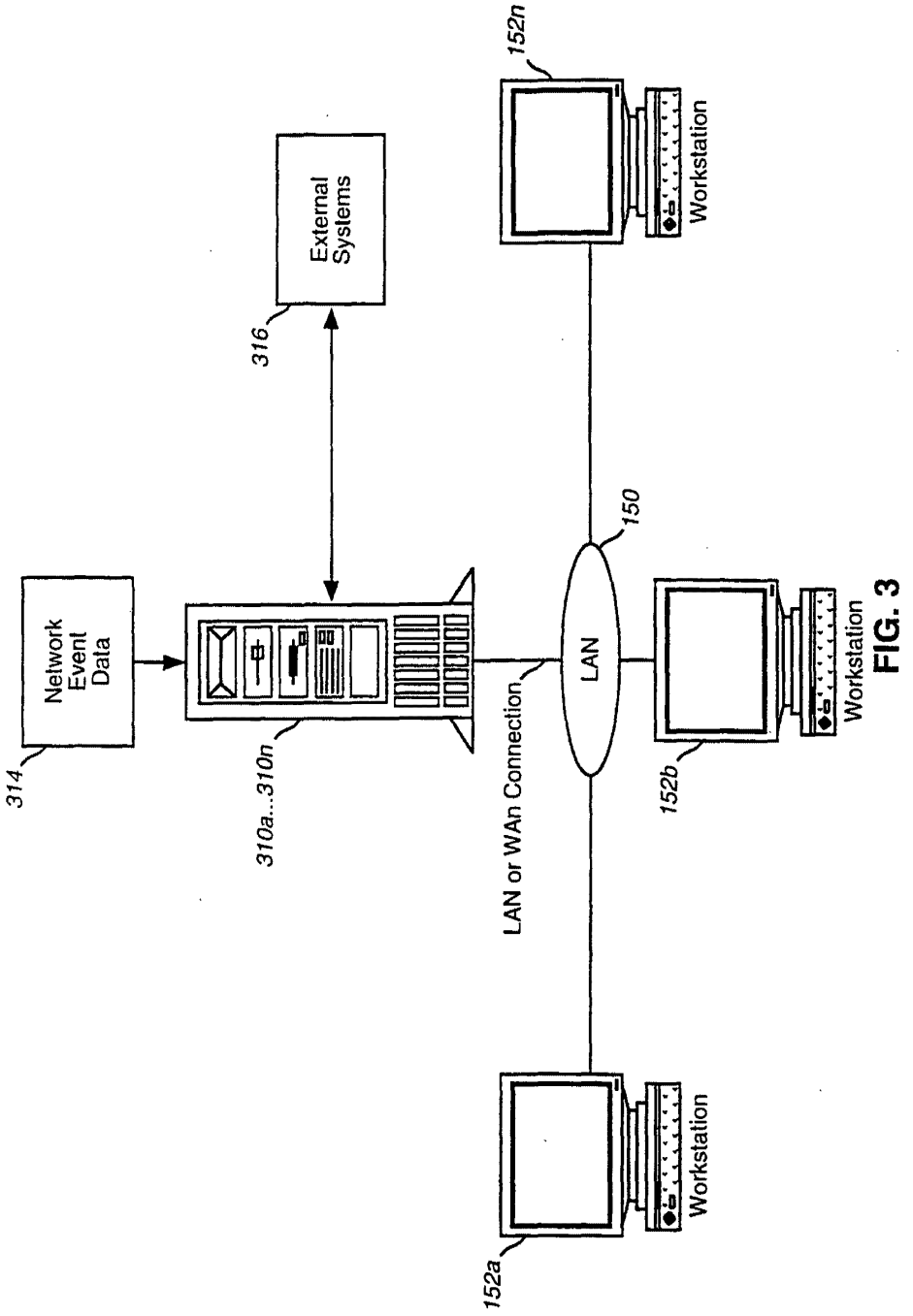


FIG. 3

IPR2024-00027
CrowdStrike Exhibit 1002 Page 430

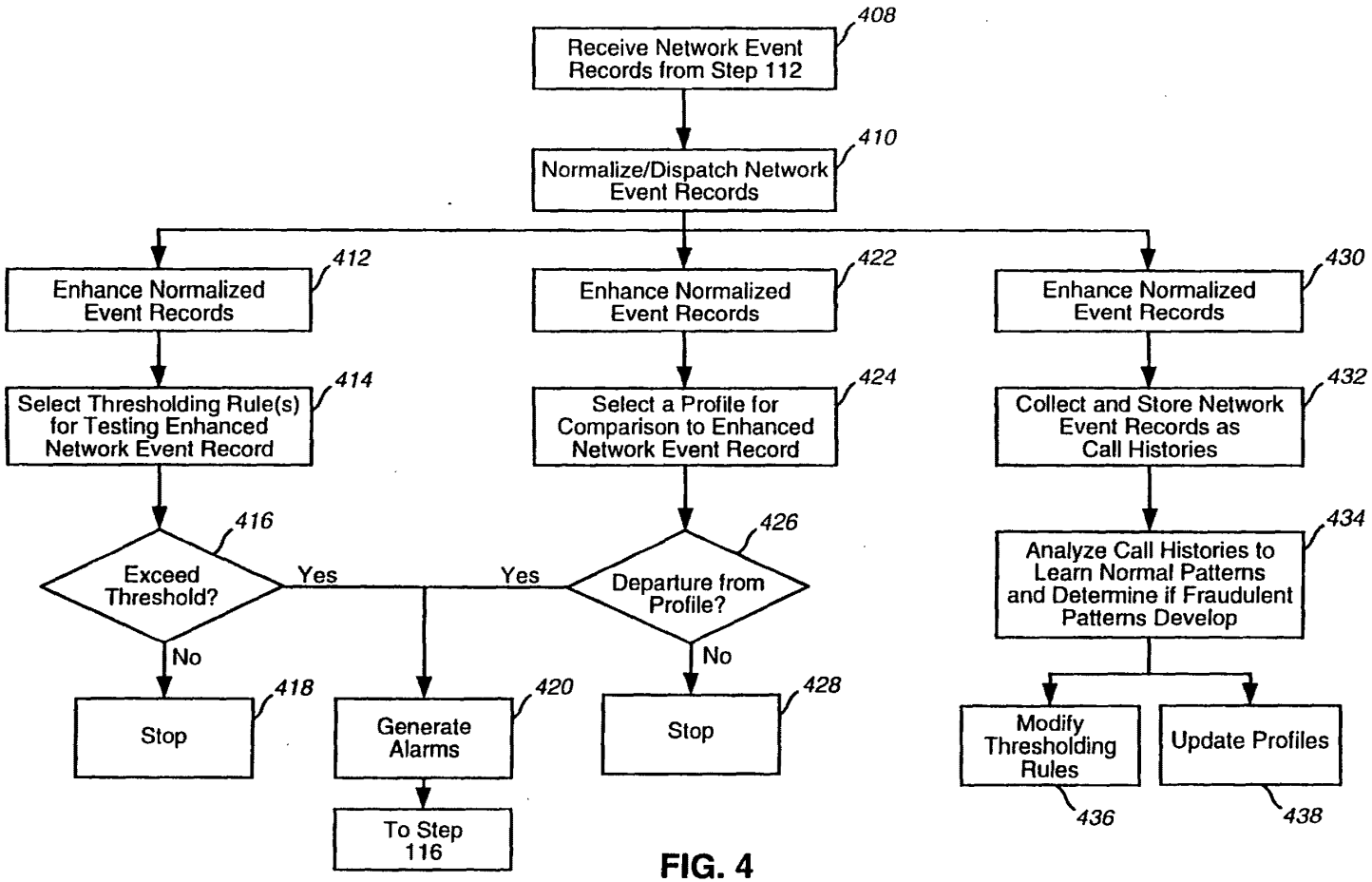

Receive Network Event Records from Step 112
408
Normalize/Dispatch Network Event Records
410
Enhance Normalized Event Records
412
Select Thresholding Rule(s) for Testing Enhanced Network Event Record
414
Exceed Threshold?
416
Yes
No
Stop
418
Generate Alarms
420
To Step 116
Enhance Normalized Event Records
422
Select a Profile for Comparison to Enhanced Network Event Record
424
Departure from Profile?
426
Yes
No
Stop
428
Enhance Normalized Event Records
430
Collect and Store Network Event Records as Call Histories
432
Analyze Call Histories to Learn Normal Patterns and Determine if Fraudulent Patterns Develop
434
Modify Thresholding Rules
436
Update Profiles
438

FIG. 4

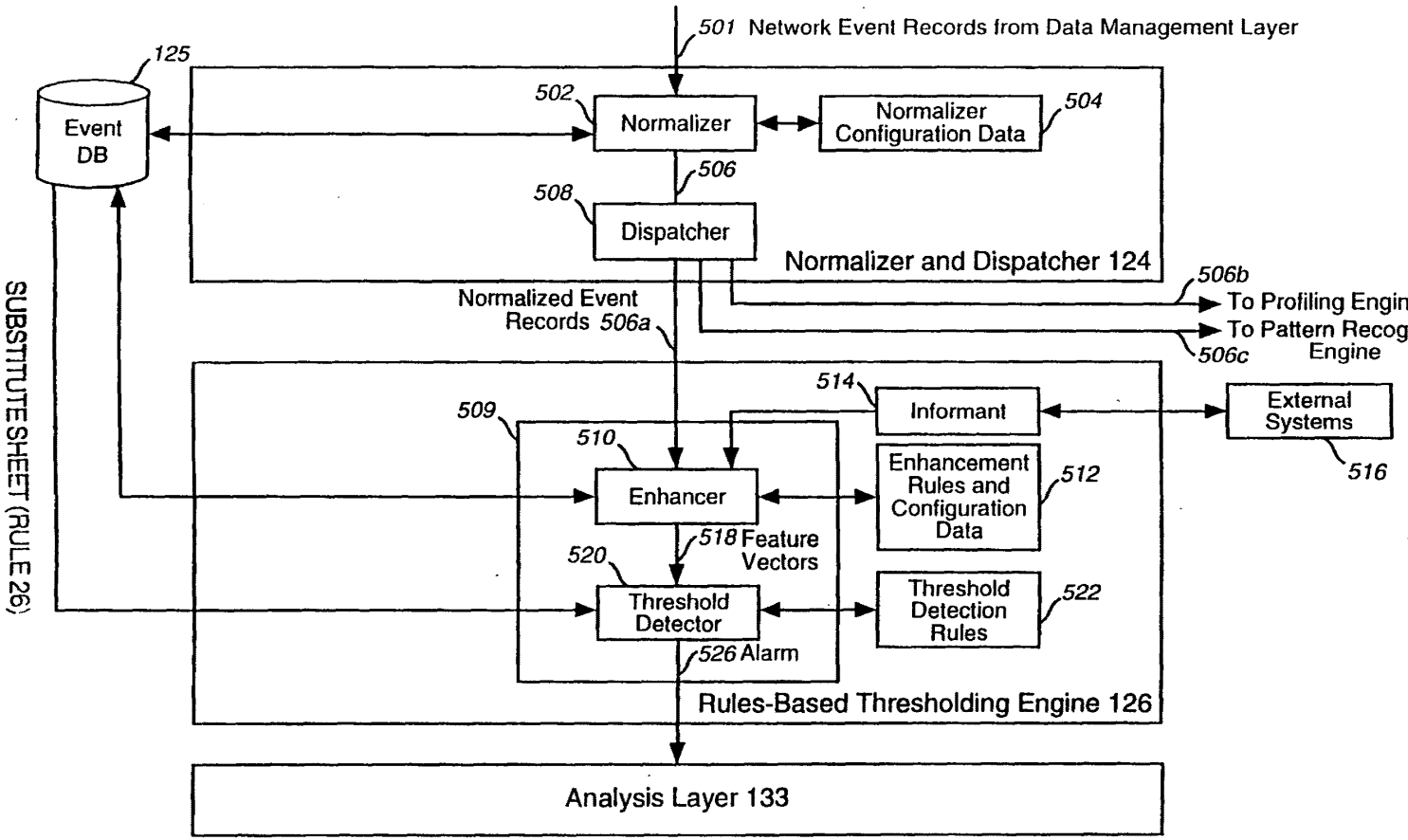


FIG. 5A

SUBSTITUTE SHEET (RULE 26)

SUBSTITUTE SHEET (RULE 26)

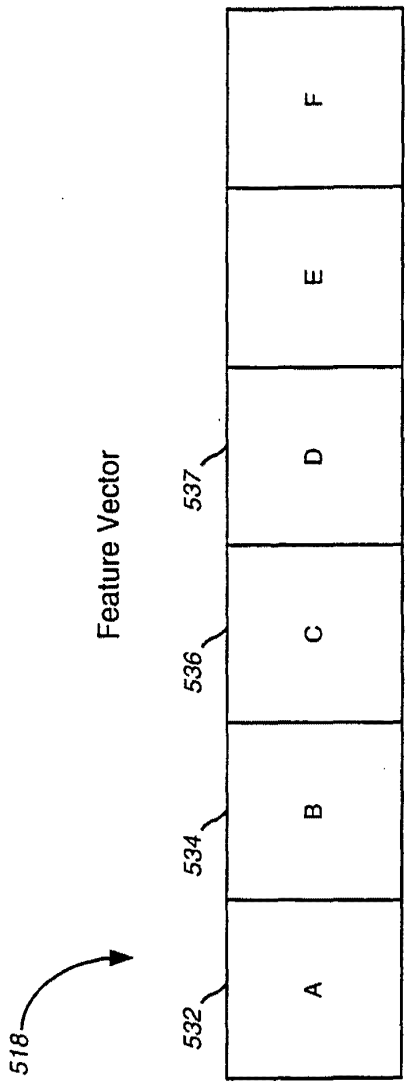


FIG. 5B

IPR2024-00027
CrowdStrike Exhibit 1002 Page 433

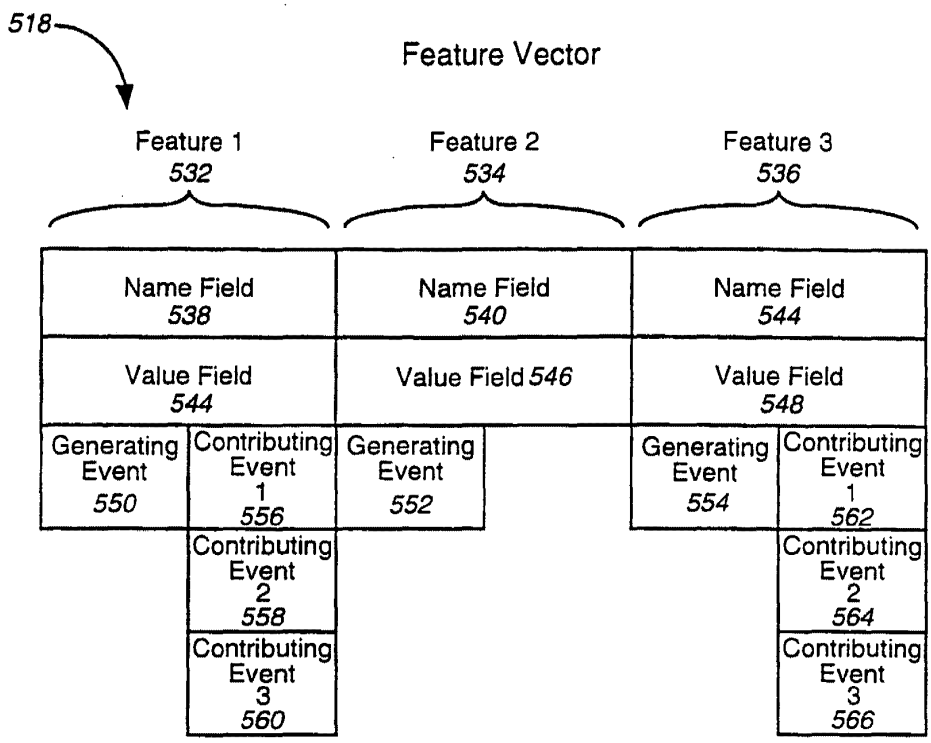


FIG. 5C

IPR2024-00027
CrowdStrike Exhibit 1002 Page 434

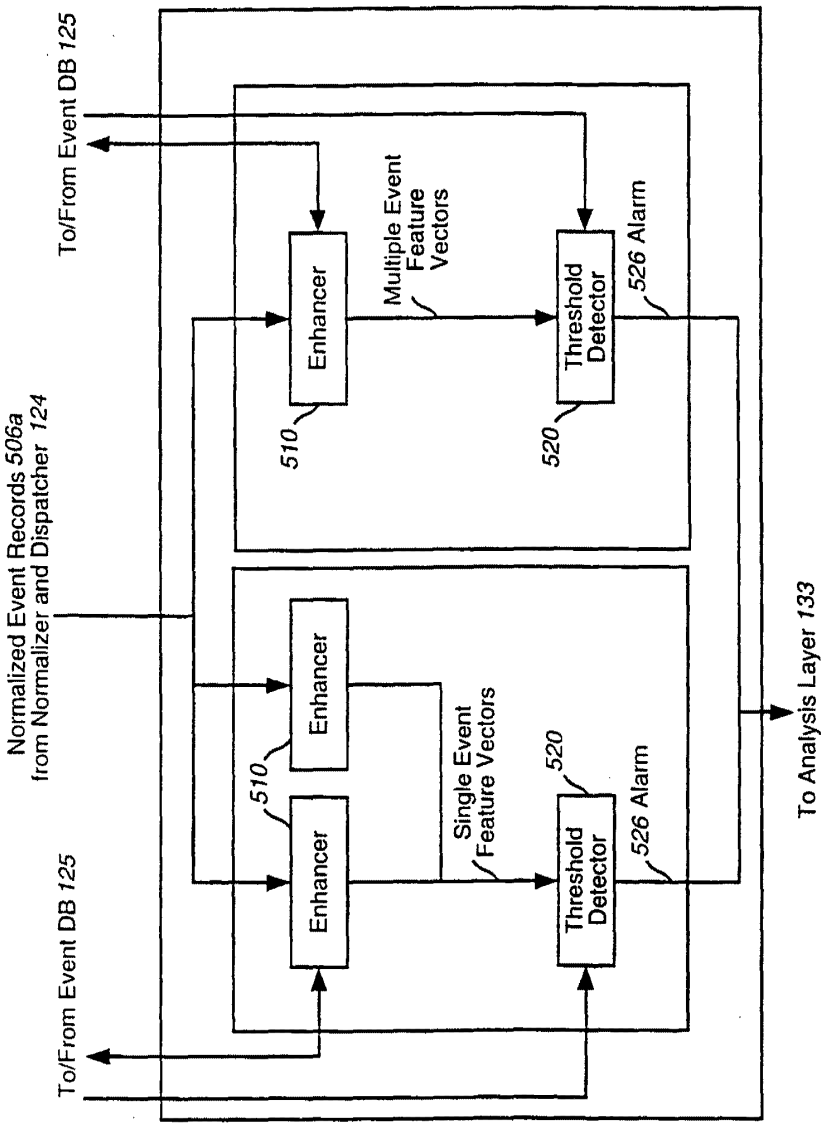


FIG. 5D

IPR2024-00027
CrowdStrike Exhibit 1002 Page 435

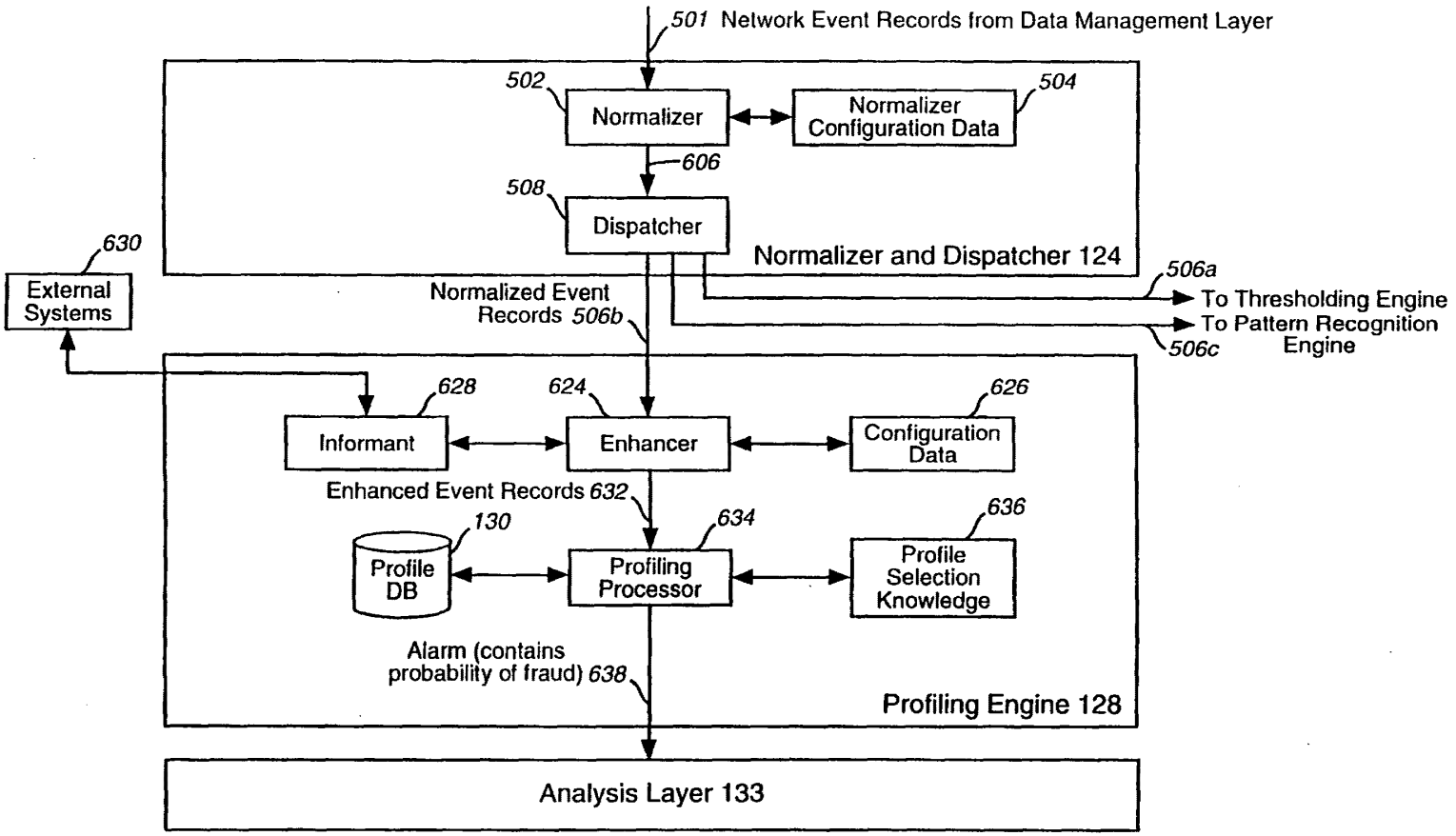


FIG. 6

SUBSTITUTE SHEET (RULE 26)

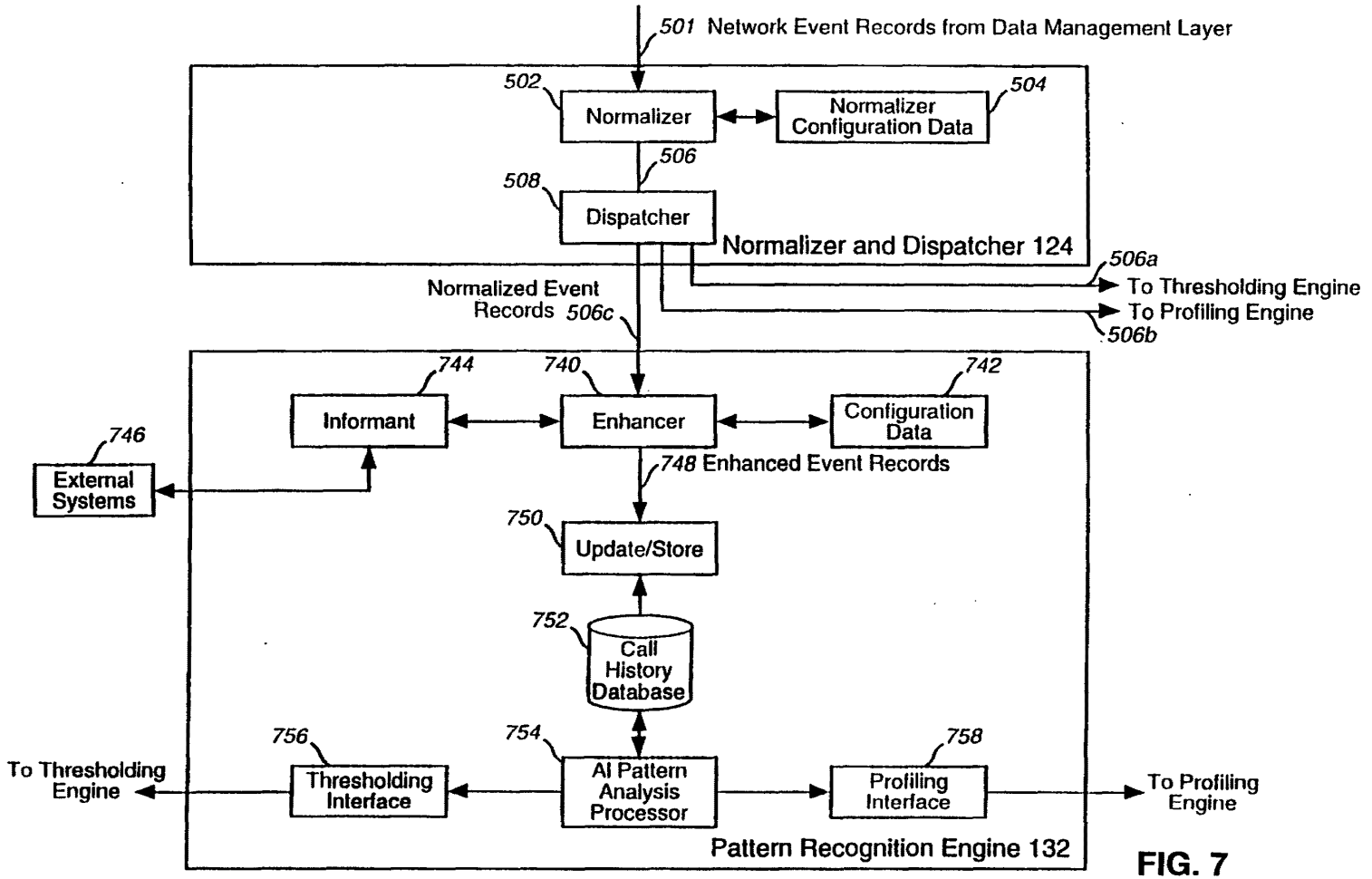
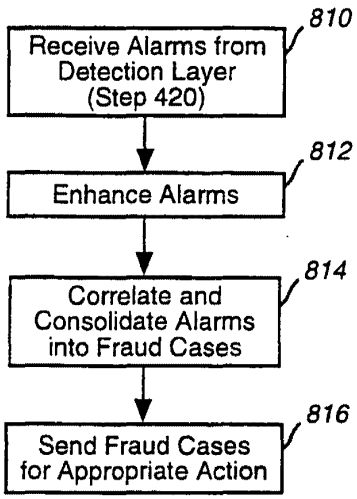

501 Network Event Records from Data Management Layer
502
Normalizer
504
Normalizer Configuration Data
506
508
Dispatcher
Normalizer and Dispatcher 124
506a
To Thresholding Engine
To Profiling Engine
506b
Normalized Event Records 506c
744
740
742
Informant
Enhancer
Configuration Data
746
External Systems
748 Enhanced Event Records
750
Update/Store
752
Call History Database
756
754
758
To Thresholding Engine
Thresholding Interface
AI Pattern Analysis Processor
Profiling Interface
To Profiling Engine
Pattern Recognition Engine 132

FIG. 7



Alarm Consolidation, Correlation and Reduction

FIG. 8

IPR2024-00027
CrowdStrike Exhibit 1002 Page 438

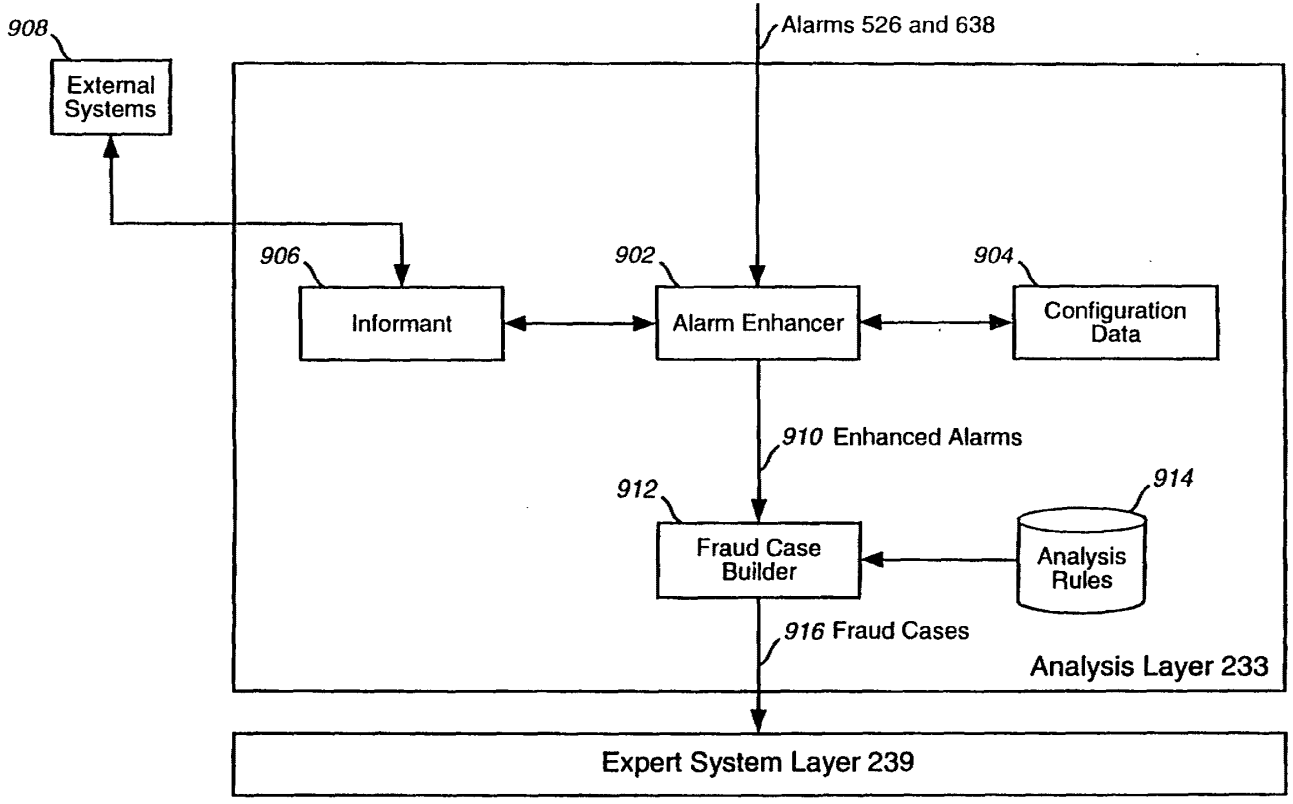
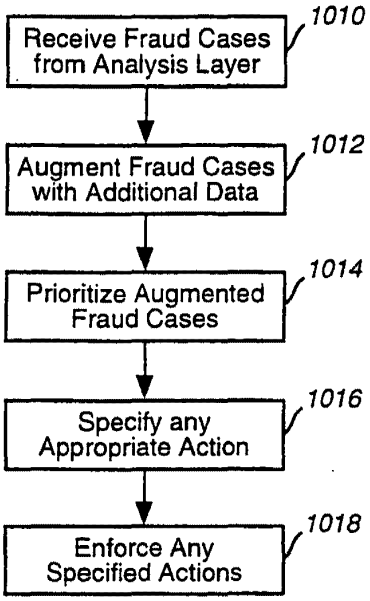


FIG. 9

SUBSTITUTE SHEET (RULE 26)

WO 99/13427 PCT/US98/19039



Fraud Case Processing in an Expert System Layer

FIG. 10

IPR2024-00027
CrowdStrike Exhibit 1002 Page 440

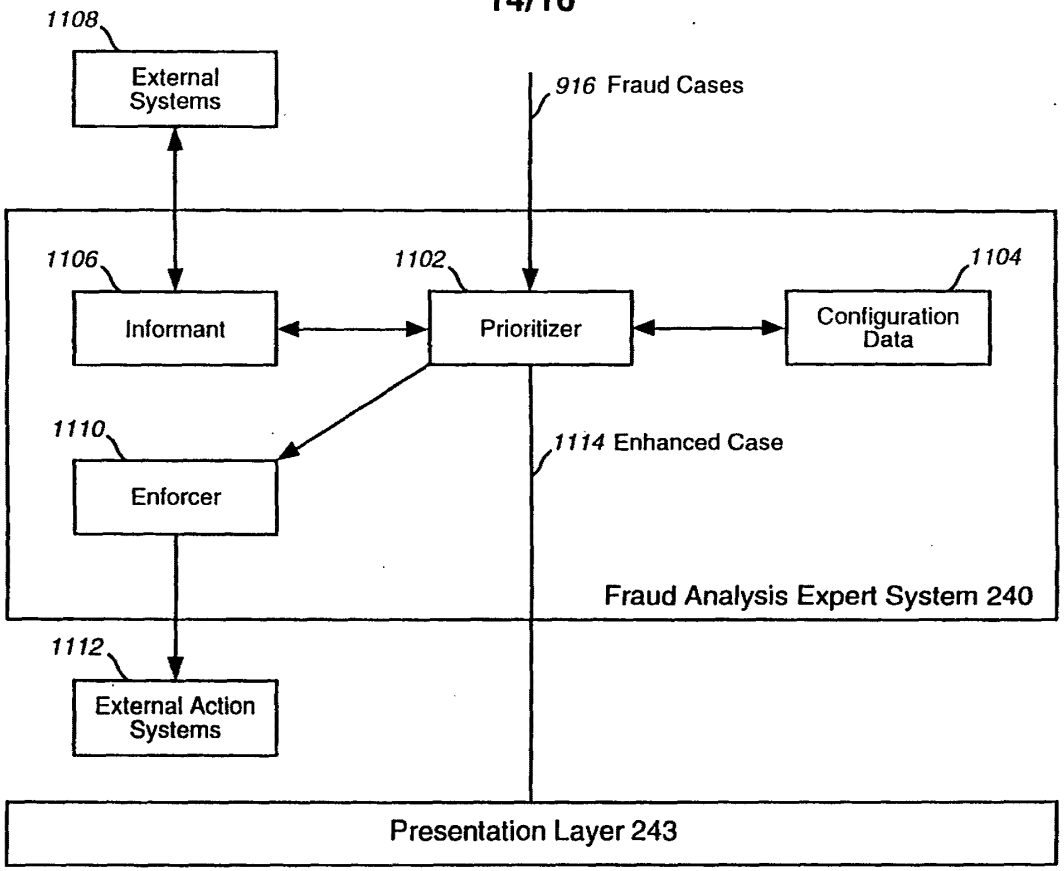


FIG. 11

WO 99/13427
PCT/US98/19039
SUBSTITUTE SHEET (RULE 26)

IPR2024-00027
CrowdStrike Exhibit 1002 Page 441

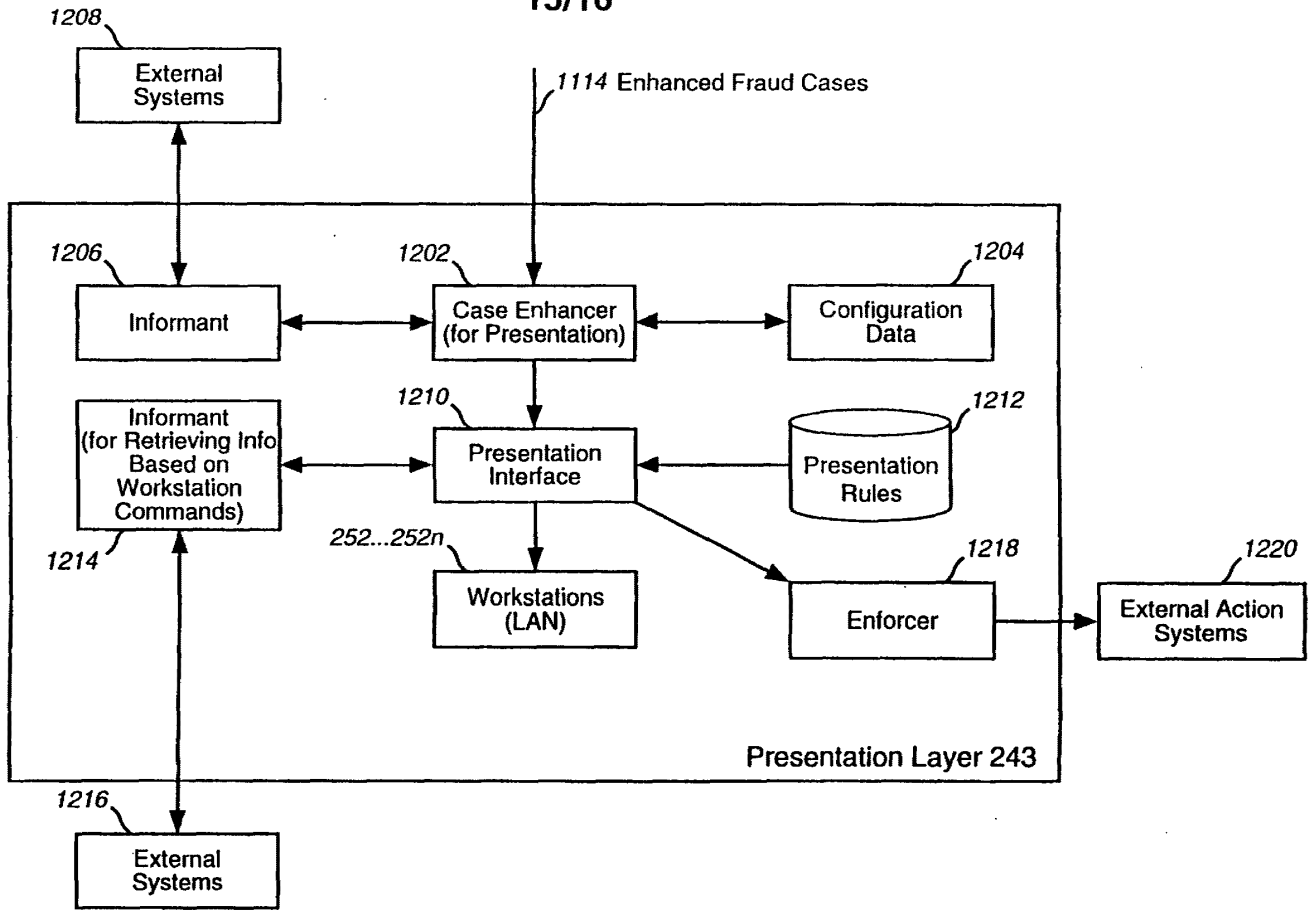


FIG. 12

SUBSTITUTE SHEET (RULE 26)

WO 99/13427 PCT/US98/19039

IPR2024-00027
CrowdStrike Exhibit 1002 Page 442

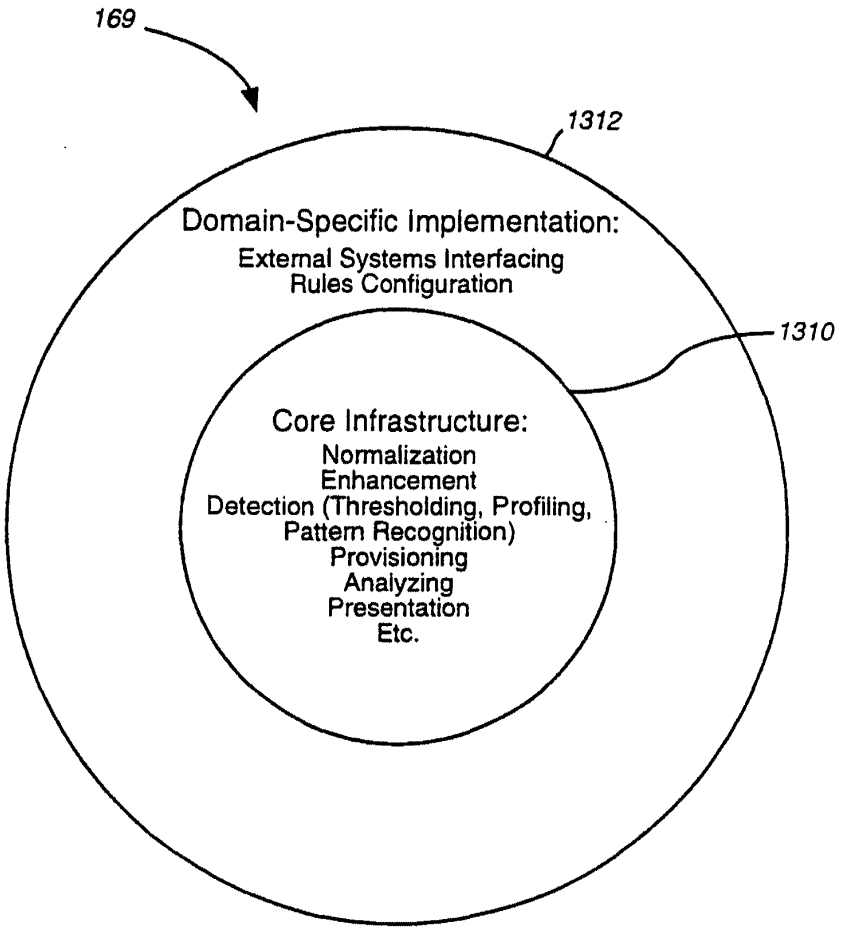


FIG. 13

IPR2024-00027
CrowdStrike Exhibit 1002 Page 443

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/16 — A2

(11) International Publication Number: **WO 99/57626**

(43) International Publication Date: 11 November 1999 (11.11.99)

(21) International Application Number: PCT/US99/09369

(22) International Filing Date: 29 April 1999 (29.04.99)

(30) Priority Data:

60/083,886	1 May 1998 (01.05.98)	US
60/092,706	14 July 1998 (14.07.98)	US
09/149,548	8 September 1998 (08.09.98)	US

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

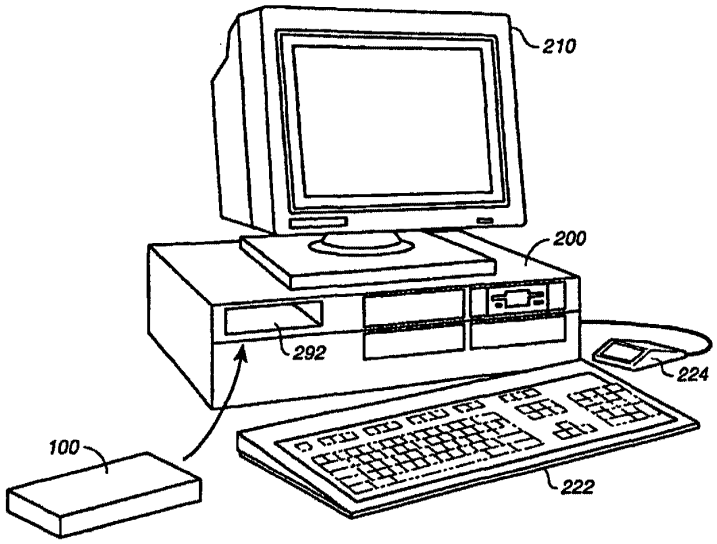
Published
Without international search report and to be republished upon receipt of that report.

(71) Applicant: ACQIS TECHNOLOGY, INC. [US/US]; 1621 West El Camino Real, Mountain View, CA 94040 (US).

(72) Inventor: CHU, William, W., Y.; 1320 Miravalle Avenue, Los Altos, CA 94024 (US).

(74) Agents: KAPOUYTIAN, Ararat et al.; Morrison & Foerster LLP, 425 Market Street, San Francisco, CA 94105-2482 (US).

(54) Title: PERSONAL COMPUTER PERIPHERAL CONSOLE WITH ATTACHED COMPUTER MODULE



(57) Abstract

An attached computer module (100) (ACM) that mates with a personal computer peripheral console (200) (PCON) to form a fully functional computer is disclosed. The ACM contains high-value components of the computer system in a compact, highly portable form. The PCON provides primary input (222, 224), output (210), and power capability. PCONs may take many forms allowing reuse of the high-value components of the computer system in different use configurations or in geographically disparate locations. The ACM and PCON may be interconnected using a cable-friendly interface with high reliability.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 444

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

PERSONAL COMPUTER PERIPHERAL CONSOLE WITH ATTACHED COMPUTER MODULE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims any and all benefits as provided by law of U.S. Provisional Application No. 60/083,886 filed May 1, 1998 and of U.S. Provisional Application No. 60/092,706 filed July 14, 1998 and of U.S. Application No. 09/149,548 filed September 8, 1998.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to the field of personal computers. In particular, the invention relates to a personal computer comprising a computing module that attaches to a mating peripheral console.

2. Description of Related Art

Most computer systems are designed as standalone, self-contained units. A personal computer (PC) is constructed with a motherboard, enclosed within a case, acting as the central circuit board that connects all devices together including the central processing unit (CPU), system memory, graphics devices, audio devices, communications devices, a power supply, hard disk drive, floppy disk drive, add-on cards, and others. While some components may be exposed to the exterior of the case for easy substitution and replacement, such as removable diskette drives or PCMCIA cards for a notebook computer, the CPU is fixed within the case. A new generation of processor "modules," such as Intel's Mobile CPU module, contain the CPU and certain support circuitry within a pluggable module, but the module is directly attached to the motherboard, enclosed within the computer case, and removed only for servicing. As such, the CPU, which is an expensive component of the computer, cannot be readily utilized apart from the system in which it is installed.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 446

Improved modular designs for personal computer systems have been suggested in the past. U.S. Patent 5,539,616 (Kikinis) shows a notebook computer comprised almost entirely of pluggable modules. This design wins the advantages most often associated with modularity, i.e., flexibility in configuration and ease of servicing. At this level of modularity, however, no single module is sufficient in itself to preserve the core computing capability and environment of the computer user.

Another approach to personal computer modularity suffers from the same shortcoming. The recently developed Device Bay standard defines a mechanism for easily adding and upgrading PC peripheral devices without opening the computer case. Device Bay supports a wide variety of storage devices. The Device Bay standard supports only peripheral devices, however, and not CPU or memory modules.

Notebook computers with docking stations represent a partitioning of PC components that permits the core computing capability and environment of the user to be isolated to a portable physical package, i.e., the notebook computer. The notebook is self-contained and fully able to operate apart from any docking station, having all core computing capability plus primary input and display devices integrated into a single package. The docking station is an optional accessory that may be used to hold secondary or bulky peripheral devices.

The portability of notebook computers is, however, constrained by several factors. As a fully functional computer system, a notebook computer requires a substantial power supply. Batteries and AC adapters are both heavy limiting the ability to produce a device that is lightweight. A notebook computer also supplies primary input and display devices for the user. Usable keyboards and readable display screens limit the ability to produce a device with small dimensions that can support the software applications most commonly used on personal computers.

The most significant partitioning of a desktop personal computer occurs in the IBM Aptiva S Series. The Aptiva S PC's incorporate a system tower with a physically separate media console connected by a bus cable. The media console contains frequently accessed peripherals, such as CD-ROM and diskette drives, and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 447

has a connection for the keyboard and mouse. This construction removes the bulky tower case from the desktop by locating a small set of low performance peripherals near the monitor, as much as six feet away from the tower. The major components of the system, including the CPU, memory, hard disk drive, add-on cards, power supply, etc., remain together in the tower case.

Consequently, there is a need in the art for a personal computer that allows the user to localize their core computing power and software environment in a small, lightweight, single, portable, physical package.

SUMMARY OF THE INVENTION

A personal computer system comprising two physically separate units and the interconnection between them is disclosed. The first unit, an attached computing module (ACM), contains the core computing power and environment for a computer user. The second unit, a peripheral console (PCON), contains the power supply and primary input and output devices for the computer system. An ACM and a PCON are coupled with one another to form a fully functional personal computer system.

The ACM is small in size so as to be easily transported between work locations or to a servicing facility. The ACM is of comparable size to that of a videocassette, and similarly has a hard shell to provide mechanical protection for its internal components. The core computing power in the ACM comprises the central processing unit (CPU), system memory, any auxiliary processors, and primary mass storage (e.g., a hard disk drive) which serves as the boot device for the computer system. The user's core environment contained in the ACM comprises the primary operating system software files, frequently used application software files, files containing the user's working data, and stored configuration data that controls various aspects of software operation customized to the user's characteristics or preferences. Notably absent from the ACM are any substantial power supply, and any substantial input/output device such as would normally be used by the computer operator to interact and exploit the range of functionality provided by the operating system and application software.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 448

The PCON provides the remaining components of a personal computer system including substantial power supply and input/output devices. Different PCON designs provide different usage possibilities for the user's core computing power and environment. Example PCON's include desktop computer, notebook computer, notepad computer, and computer-based entertainment computer designs.

To form a fully operational computer system, an ACM is coupled with a PCON. The plug-in module design of the ACM, and the concentration of high-value components therein (both in terms of high-value hardware and high-value files), makes it easy for a user to transport the high-value core between multiple PCON's, each of which can enjoy a relatively low cost. The concentration of a user's core computing environment in a small, portable package also makes it possible for large organizations to perform moves, adds, and changes to personal computer systems with greater efficiency.

These and other purposes and advantages of the present invention will become more apparent to those skilled in the art from the following detailed description in conjunction with the appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 depicts an exemplary desktop peripheral console and attached computing module.

Figures 2a through **2d** depict various peripheral console configurations.

Figure 3 is a block diagram of one embodiment of a computer system employing the present invention.

Figure 4 is a block diagram of an attached computing module (ACM).

Figure 5 illustrates an external view of one embodiment of an ACM.

Figure 5b illustrates one possible embodiment of a computer bay.

Figure 6 illustrates the internal component layout for one embodiment of an ACM.

Figure 7 is a block diagram of a peripheral console (PCON).

Figure 8 depicts internal major component placement for one tower desktop peripheral console (PCON).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 449

Figure 9 depicts internal component placement for one notebook peripheral console (PCON).

In the figures just described, like parts appearing in multiple figures are numbered the same in each figure.

DETAILED DESCRIPTION

Figure 1 depicts an exemplary desktop peripheral console and attached computing module. The desktop peripheral console (PCON) looks similar to a desktop PC system unit of conventional design. Front covers for device bays and a diskette drive are visible on the front panel of the PCON. The PCON also provides connections for a display monitor, a keyboard, and a mouse.

The front panel of the PCON also displays an opening to a computer bay 292. The computer bay acts as the receptacle for the attached computer module (ACM). The ACM houses the core computing power and environment for a particular user and is inserted into the opening of the computer bay to receive power and to interact with the peripheral devices housed in the PCON. The ACM achieves normal operational capability when mated with a PCON. Because the ACM does not contain a primary power supply or primary input or output devices, it can be small and lightweight. These characteristics make the ACM greatly portable. It can be easily carried in a briefcase with other matter and is thus ideal for transport between home and office, or multiple office locations, each equipped with a desktop PCON. This represents one advantage of the present invention.

The design of peripheral consoles (PCON's) is in no way limited to the desktop unit as pictured in **Figure 1**. **Figures 2a** through **2d** depict various peripheral console configurations. **Figure 2a** depicts a tower desktop PCON configuration. The opening of the computer bay 292 is visible at the front of the PCON unit. The PCON provides support for a video monitor as the user's primary display device. The PCON also provides support for a keyboard and a mouse as the user's primary input (text and pointing) devices.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 450

Figure 2b depicts a notebook computer PCON configuration. The opening of the computer bay 292 is visible at the side of the PCON unit. The PCON provides an integrated LCD display panel as the user's primary display device. The PCON provides an integrated keyboard as the user's primary input device.

Figure 2c depicts a notepad computer PCON configuration. The opening of the computer bay 292 is visible along the back side of the PCON unit. The PCON provides an integrated LCD display panel as the user's primary display device. The display panel may be equipped with touch sensitivity to serve as the user's primary input device. The stylus may be used to enter text or graphics, or to select "soft" buttons, on the touch sensitive screen. Software accessible mechanical switches serve as an alternative primary input mechanism.

Figure 2d depicts an entertainment PCON configuration. The opening of the computer bay 292 is visible at the front side of the PCON unit. The PCON provides an integrated television screen as the user's primary display device. A remote control keypad serves as the user's primary input device.

Figure 3 is a block diagram of the components in one computer system employing the present invention. The computer system comprises an attached computer module (ACM), a peripheral console (PCON), and the interconnection apparatus between them. The ACM includes the central processing unit (CPU) 110, system memory 120, high performance devices 150, primary mass storage 130, and related interface and support circuitry 140. The PCON includes primary display 210, primary input 220, secondary mass storage 250, other devices 260, expansion slots 270, the primary power supply 230, and related interface and support circuitry 240. The interconnection apparatus 300 includes circuitry to convey power and operational signals between the ACM and PCON.

Within the ACM 100, the CPU 110 executes instructions and manipulates data stored in the system memory. The CPU 110 and system memory 120 represent the user's core computing power. The core computing power may also include high performance devices 150 such as advanced graphics processor chips that greatly increase overall system performance and which, because of their speed, need to be located close to the CPU. The primary mass storage 130 contains

IPR2024-00027
CrowdStrike Exhibit 1002 Page 451

persistent copies of the operating system software, application software, configuration data, and user data. The software and data stored in the primary mass storage device represent the user's computing environment. Interface and support circuitry 140 primarily includes interface chips and signal busses that interconnect the CPU, system memory, high performance devices, and primary mass storage. The interface and support circuitry also connects ACM-resident components with the ACM-to-PCON interconnection apparatus as needed.

Within the PCON 200, the primary display component 210 may include an integrated display device or connection circuitry for an external display device. This primary display device may be, for example, an LCD, plasma, or CRT display screen used to display text and graphics to the user for interaction with the operating system and application software. The primary display component is the primary output of the computer system, i.e., the paramount vehicle by which programs executing on the CPU can communicate toward the user.

The primary input component 220 of the PCON may include an integrated input device or connection circuitry for attachment to an external input device. The primary input may be, for example, a keyboard, touch screen, keypad, mouse, trackball, digitizing pad, or some combination thereof to enable the user to interact with the operating system and application software. The primary input component is the paramount vehicle by which programs executing on the CPU receive signals from the user.

The PCON may contain secondary mass storage 250 to provide additional high capacity storage for data and software. Secondary mass storage may have fixed or removable media and may include, for example, devices such as diskette drives, hard disks, CD-ROM drives, DVD drives, and tape drives.

The PCON may be enhanced with additional capability through the use of integrated "Other Devices" 260 or add-on cards inserted into the PCON's expansion slots 270. Examples of additional capability include sound generators, LAN connections, and modems. Interface and support circuitry 240 primarily includes interface chips, driver chips, and signal busses that interconnect the other components within the PCON. The interface and support circuitry also connects

IPR2024-00027
CrowdStrike Exhibit 1002 Page 452

PCON-resident components with the ACM-to-PCON interconnection apparatus as needed.

Importantly, the PCON houses the primary power supply 230. The primary power supply has sufficient capacity to power both the PCON and the ACM 100 for normal operation. Note that the ACM may include a secondary "power supply" in the form, for example, of a small battery. Such a power supply would be included in the ACM to maintain, for example, a time-of-day clock, configuration settings when the ACM is not attached to a PCON, or machine state when moving an active ACM immediately from one PCON to another. The total energy stored in such a battery would, however, be insufficient to sustain operation of the CPU at its rated speed, along with the memory and primary mass storage, for more than a fraction of an hour, if the battery were able to deliver the required level of electrical current at all.

Figure 4 is a block diagram of an attached computing module (ACM) 100. The physical ACM package 100 contains the ACM functional components 101 and the ACM side of the ACM-to-PCON Interconnection 300. The ACM 101 comprises a CPU component 110, a system memory component 120, a primary mass storage component 130, a high performance devices components 150, and an interface and support component 140.

The ACM side of the ACM-to-PCON Interconnection 300 comprises a Host Interface Controller (HIC) component 320 and an ACM connector component 330. The HIC 320 and connector 330 components couple the ACM functional components 101 with the signals of an ACM-to-PCON interface bus 310 used to operatively connect an ACM with a PCON. The ACM-to-PCON interface bus 310 comprises conveyance for electrical power 314 and signals for a peripheral bus 312, video 316, video port 317, and console type 318.

The ACM-to-PCON Interconnection 300 conveys certain signals in bit-serial format. The use of bit-serial format reduces the number of physical signal paths required to traverse the interconnection. Additionally, the ACM-to-PCON Interconnection 300 uses differential signaling to convey certain bit-serial data.. Differential signaling, such as low voltage differential signaling (LVDS), can

IPR2024-00027
CrowdStrike Exhibit 1002 Page 453

provide very reliable, high-speed, and low-power transmission across cables -- and cables as a transmission media provide tremendous design flexibility. Accordingly, the use of bit-serial format for computer bus data and the use of differential signaling represent further advantages of the present invention.

More specifically, in the presently described embodiment at least the peripheral bus 312 has a plurality of serial bit channels numbering fewer than the number of parallel bus lines in PCI bus 167 and operates at a clock speed higher than the clock speed at which any of the PCI bus lines operates. The peripheral bus 312 includes two sets of unidirectional serial bit channels which transmit data in opposite directions such that one set of bit channels conveys serial bits from the HIC while the other set conveys serial bits to the HIC. For each cycle of the PCI bus 167 clock, each bit channel of the peripheral bus 312 transmits a packet of multiple serial bits.

HIC 320 includes a number of functional subsections that are not shown individually in **Figure 4**. The HIC includes a bus controller to interface with PCI bus 167 and to manage transactions that occur therewith. The HIC also includes a translator coupled to the bus controller to (1) encode control signals from PCI bus 167 into control bits and to decode control bits conveyed by peripheral bus 312 into generated control signals for PCI bus 167, and (2) to manage the movement of data and address bit information to and from PCI bus 167. Additionally, HIC 320 includes a transmitter and a receiver, each coupled to the translator. The transmitter converts parallel bits representing data, address, and control signals into serial bits and transmits the serial bits to the peripheral bus 312 using differential signaling. The receiver receives serial bits from a differential signal of peripheral bus 312, and converts the serial bits representing data, address, and control signals into parallel bits.

The CPU component 110 of the ACM functional circuitry 101 of the presently described embodiment comprises a microprocessor 112, which is the chief component of the personal computer system, power supply connection point 113, and cache memory 114 tightly coupled to the microprocessor 112 by the CPU-to-cache bus 174 comprising signal paths for address, data, and control information.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 454

The microprocessor 112 of this embodiment is one of the models from the Pentium II family of processors from Intel Corporation. Microprocessor 112 receives electrical power from power bus 168 via connection point 113. Microprocessor 112 couples to the Host Interface Controller (HIC) 320 via CPU-to-HIC bus 163 comprising signal paths to exchange control information such as an interrupt request. Microprocessor 112 also couples to CPU Bridge 146 via CPU main bus 164 comprising signal paths for address, data, and control information.

The CPU Bridge component 146 of the interface and support circuitry 140 operates to couple the high speed CPU main bus 164 to specialty buses of varying speeds and capability that connect other computer components. The CPU Bridge of the presently described embodiment incorporates memory controller circuitry, advanced graphics processor support circuitry, and a general, industry-standard PCI bus controller in a single package. A CPU Bridge 146 such as the 82443LX PCI/AGP Controller from Intel Corporation may be used.

The system memory component 120 of the ACM functional circuitry 101 in the present embodiment comprises main system memory (RAM) 122, BIOS memory 124, and flash memory 126. The system memory 120 is used to contain data and instructions that are directly addressable by the CPU. The RAM 122 comprises volatile memory devices such as DRAM or SDRAM memory chips that do not retain their stored contents when power is removed. This form of memory represents the largest proportion of total system memory 120 capacity. The BIOS memory 124 comprises non-volatile memory devices such as ROM or EPROM memory chips that retain their stored contents regardless of the application of power and are read-only memory under normal operating conditions. The BIOS memory 124 stores, for example, start-up instructions for the microprocessor 112 and sets of instructions for rudimentary input/output tasks. The flash memory 126 comprises non-volatile memory devices that retain their stored contents regardless of the application of power. Unlike the BIOS non-volatile memory, however, the stored contents of the flash memory 126 are easily changed under normal operating conditions. The flash memory 126 may be used to store status and configuration data, such as security identifiers or ACM specifications like the speed of the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 455

microprocessor 112. Some embodiments may combine the BIOS functions into the flash memory device, thus permitting BIOS contents to be rewritten, improving field upgradability.

The main system memory (RAM) 122 is coupled to memory controller circuitry resident within the CPU Bridge 146 via direct memory bus 165. The BIOS 124 and flash memory 126 are coupled to HIC 320 via switched memory bus 166. This permits the BIOS 124 and flash 126 memories to be accessed by circuitry in the HIC 320 or other circuitry connected thereto. The direct memory bus 165 and the switch memory bus 166 each comprises conductors to convey signals for data, address, and control information.

The primary mass storage component 130 of the ACM functional circuitry 101 in the present embodiment comprises a compact hard disk drive with an industry-standard, IDE interface. The hard disk drive (HDD) 132 has a formatted storage capacity sufficient to contain an operating system for the computer, application software desired by the user, and related user configuration and operating parameter data. The HDD 132 in the present embodiment serves as the "boot" device for the personal computer from which the operating system is loaded into RAM 122 by the start-up program stored in the BIOS 124.

The present HDD 132 has a capacity of approximately 2,000 megabytes to provide adequate storage for common software configurations and reasonable space for user data. One example of a common software configuration includes the Windows 95 operating system from Microsoft Corporation, a word processing program, a spreadsheet program, a presentation graphics program, a database program, an email program, and a web browser such as Navigator from Netscape Corporation. The hard disk 132 stores program and data files for each software component, including files distributed by the vendor as well as files created or updated by operation of the software after it is installed. For example, a word processor program may maintain information about a user's identity and latest preferences in an operating system registry file. Or, for example, the web browser may maintain a file of the user's favorite web sites or most recently viewed web pages. An HDD with 2000 megabyte capacity is readily available in the small size

IPR2024-00027
CrowdStrike Exhibit 1002 Page 456

of hard disk (e.g., 2.5-inch or 3.5-inch) to minimize the space required within the ACM for the primary mass storage device 130.

The HDD 132 is coupled to IDE controller circuitry 148 via IDE bus 172. The IDE controller circuitry 148 is coupled to the CPU Bridge 146 via the Host PCI bus 167. IDE controllers and busses, and the PCI bus are well known and understood in the industry. The above components operate together to couple the hard disk drive 132 to the microprocessor 112.

The high performance devices component 150 of the ACM functional circuitry 101 in the present embodiment comprises an Advanced Graphics Processor (AGP) 152. The Model 740 Graphics Device from Intel Corporation may be used in the present embodiment as the AGP.

Increases in computer screen size, graphics resolution, color depth, and visual motion frame rates, used by operating system and application software alike, have increased the computing power required to generate and maintain computer screen displays. An AGP removes a substantial portion of the graphics computing burden from the CPU to the specialized high-performance processor, but a high level of interaction between the CPU and the specialized processor is nonetheless required. To maximize the effective contribution of having a specialized processor in the presently described embodiment, the AGP 152 is located in the ACM 100, where it is in close proximity to the microprocessor 112. The AGP 152 is coupled to the microprocessor 112 via the advanced graphics port bus 173 of the CPU Bridge 146. The visual display signal generated by the AGP are conveyed toward actual display devices at the peripheral console (PCON) via video signal bus 170. Video information from a source external to the ACM and appearing as video port signals 317 may be conveyed to the AGP 152 via video port signal path 171.

Other types of high performance components may be included in different ACM configurations. For example, an interface to an extremely high speed data communication facility may be desirable in some future computer where CPU-to-network interaction is of comparable intensity to today's CPU-to-graphics interaction. Because such high performance components tend to be high in cost, their inclusion in the ACM is desirable. Inclusion of high cost, high performance

IPR2024-00027
CrowdStrike Exhibit 1002 Page 457

components in the ACM concentrates a user's core computing power and environment in a portable package. This represents a further advantage of the invention.

The interface and support component 140 of the ACM functional circuitry 101 in the present embodiment comprises circuitry for power regulation 142, clocking 144, CPU Bridge 146, IDE controller 148, and signal conveyance paths 161-174. The CPU Bridge 146 couples the CPU component 110 of the ACM 100 with the other components of the ACM 120-150 and the CPU-to-PCON Interconnection 300. The CPU Bridge 146 and IDE controller 148 have already been discussed. Power regulation circuitry 142 receives electrical power via the electrical power conduction path 314 of the CPU-to-PCON Interconnection 300, conditions and distributes it to the other circuitry in the ACM using power distribution bus 168. Such regulation and distribution is well known and understood in the art.

Clocking circuitry 144 generates clock signals for distribution to other components within the ACM 100 that require a timing and synchronization clock source. The CPU 110 is one such component. Often, the total power dissipated by a CPU is directly proportional to the frequency of its main clock signal. The presently described embodiment of the ACM 100 includes circuitry that can vary the frequency of the main CPU clock signal conveyed to the CPU via signal path 162, in response to a signal received from the host interface controller (HIC) 320 via signal path 161. The generation and variable frequency control of clocking signals is well understood in the art. By varying the frequency, the power consumption of the CPU (and thus the entire ACM) can be varied.

The variable clock rate generation may be exploited to match the CPU power consumption to the available electrical power. Circuitry in the host interface controller (HIC) 320 of the presently described embodiment adjusts the frequency control signal sent via signal path 161 to the clocking circuitry 144, based on the "console type" information signal 318 conveyed from the peripheral console (PCON) by the CPU-to-PCON interconnection 300. In this arrangement, the console type signal originating from a desktop PCON, such as depicted in **Figure**

IPR2024-00027
CrowdStrike Exhibit 1002 Page 458

2a, would result in the generation of a maximum speed CPU clock. The desktop PCON, presumably has unlimited power from an electrical wall outlet and does not need to sacrifice speed for power conservation. The console type signal originating from a notebook PCON, such as depicted in **Figure 2b**, would, however, result in the generation of a CPU clock speed reduced from the maximum in order to conserve battery power and extend the duration of computer operation obtained from the energy stored in the battery. The console type signal originating from a notepad PCON, such as depicted in **Figure 2c**, would result in the generation of a CPU clock speed reduced further yet, the notepad PCON presumably having smaller batteries than the notebook PCON. Inclusion of control signals and circuitry to effect a CPU clock signal varying in frequency according to characteristics of the PCON to which the ACM is connected facilitates the movement of the user's core computing power and environment to different work settings, which is a further advantage of the present invention.

Figure 5 illustrates an external view of one embodiment of an ACM. The case 510 of the ACM 100 is generally rectangular in shape, preferably constructed of a strong, lightweight, rigid material that will protect the internal components from mechanical and environmental exposure. Plastics may readily be used to construct the case 510. The case 510 completely surrounds the internal components, being generally an 6-sided box. **Figure 5** shows the top 512, right 514, and rear 516 surfaces of the ACM case 510. Rear edges 518 of the case joining the rear surface 516 with its adjoining surfaces may be beveled or rounded to facilitate insertion of the ACM 100 into the computer bay of the PCON. Notches 540 may be formed by projecting small surfaces inward from otherwise generally flat surfaces of the ACM case 510. The notches 540 may be used to engage with mechanical devices mounted in and about a computer bay. Such mechanical devices can be employed to secure the ACM into position within a computer bay for reliability and security. Openings 517 are formed into the rear surface 516 of the ACM case 510 through which to project connectors 330a and 330b. In one embodiment the case 510 is approximately 5.75 inches wide by 6.5 inches deep by 1.6 inches high.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 459

Connectors 330a and 330b are part of the ACM-to-PCON Interconnection as described earlier in reference to **Figures 3 and 4**. When the ACM 100 is inserted into the computer bay of a peripheral console (PCON), connectors 330a and 330b mate with corresponding connectors located at the rear of the computer bay to electrically couple the ACM with the PCON containing the computer bay. The connectors 330a and 330b used in one embodiment are connectors complying with the Device Bay industry standard as documented in "Device Bay Interface Specification," revision 0.85, February 6, 1998. Such connectors have specifically been designed to stand up to the rigors of repeated insertion and withdrawal.

Cooling plate 530 forms part of the top surface 512 of ACM 100. The cooling plate 530 may be mounted to, or project through an opening formed in, case 510. Similarly, electromagnetic interference (EMI) / electrostatic discharge (ESD) grounding plate 532 forms part of the right surface 514 of ACM 100. The grounding plate 532 may be mounted to, or project through an opening formed in, case 510. Cooling plate 530 and grounding plate 532 compressively mate with counterparts when the ACM is fully inserted into the computer bay. The counterparts located along the boundaries of the computer bay conduct dangerous heat and electrical charges away from the ACM. Inside the ACM, cooling plate 530 thermally couples to heat-sensitive components such as CPU 110 by methods well known in the art. Similarly, grounding plate 532 electrically couples to EMI/ESD-sensitive components, such as a microprocessor, by methods well known in the art.

LCD display 550 forms part of the right surface 514 of ACM 100. The LCD display may be mounted to, or project through an opening formed in, case 510. The LCD display may contain indicators about the status of the ACM. Such indicators may display, for example, the time-of-day from a time-of-day clock contained within the ACM, or the amount of charge remaining in an ACM-resident battery, or certain configuration options recorded in flash memory. The LCD display 550 provides display capability for a limited amount of information, most useful when the ACM is separated from a PCON (and is thus separated from a full-capability, primary display device).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 460

Figure 5b illustrates one possible embodiment of a computer bay. A computer bay 290 acts as a receptacle for lodging an ACM (such as the one shown in **Figure 5**) within a desktop PCON. The illustrated computer bay 290 provides an ACM with housing and with signal flow, electrical grounding, heat transfer, and mechanical connections. While many physical arrangements between the ACM and PCON are possible, the use of an enclosed computer bay as the one illustrated in **Figure 5b** offers many advantages. For example, the illustrated computer bay 290 provides physical protection for the ACM. The computer bay may also be easily incorporated into industry standard form factors used in the manufacture of desktop personal computers (e.g., the ACM and associated computer bay could be designed to fit within the volume occupied by a standard-size disk drive).

The computer bay 290 appearing in **Figure 5b** is shown mounted within the confines of PCON case 202. The computer bay 290 comprises frame 291 and signal flow, grounding, cooling, and locking components as described below. Mounting flanges 298 of frame 291 may be used to attach the computer bay 290 to the PCON structure. The computer bay 290 is prominently defined by frame 291 generally forming a cavity in which to lodge an ACM. As such, the interior cavity formed by frame 291 closely approximates the exterior dimensions of a compatible ACM. The top 293, right 294, and rear 295 sides of the computer bay frame 291 are visible. The computer bay frame 291 also includes substantial bottom and left sides which are not shown. The front side of the frame 291 (not shown) is open to allow the insertion of the ACM. Frame 291 is constructed of metal for strength and to facilitate the conductance of heat and undesired electrical currents away from the ACM.

In the presently described embodiment, the weight of an inserted ACM is largely borne by the bottom side (not shown) of computer bay frame 291. Alternative embodiments are possible where, for example, the weight of the ACM is borne by rails running longitudinally down the right and left sides of the computer bay cavity that engage corresponding grooves running longitudinally down the right and left sides of an ACM.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 461

The computer bay 290 includes a signal conductor component to carry electrical signals between the circuitry in the ACM and the circuitry in the PCON. The signal conductor component comprises connectors 362a and 362b and cables 364. The signal conductor component of the computer bay 290 is logically part of the interconnection apparatus represented in block 300 of **Figure 3**.

The rear sections of connectors 362a and 362b are visible in **Figure 5b**. When an ACM is operatively inserted into the computer bay 290, connectors 362a and 362b mate with corresponding connectors located at the rear of the ACM (e.g., connectors 330a and 330b of **Figure 5**). The signals present on the pins of an inserted ACM's connectors are conducted to the corresponding pins of connectors 362a and 362b, and along corresponding conductors within cables 364, to other circuitry housed within PCON case 201. The connectors 362a and 362b used in one embodiment are connectors complying with the Device Bay industry standard as documented in "Device Bay Interface Specification," revision 0.85, February 6, 1998. Such connectors have specifically been designed to stand up to the rigors of repeated insertion and withdrawal.

The computer bay 290 includes an electrical grounding component comprising spring contact bands 620, opening 622, and a conductive path to a grounding source. The electrical grounding component of the computer bay 290 provides a mating mechanism for the grounding pad 532 (shown in **Figure 5**) on an inserted ACM. As such, the position of opening 622 and spring contact bands 620 corresponds to the position of grounding plate 532 (shown in **Figure 5**) for an ACM fully inserted into the computer bay. The bands 620 attach at their ends to side 294 of the computer bay 290 and are formed to project through opening 622 in the side 294 of the computer bay 290. The bands 620 project far enough through opening 622 so as to extend somewhat into the interior of the cavity into which an ACM is inserted. Spring contact bands 620 are constructed of an elastic, conductive material such as spring steel with a conductive coating. With an ACM inserted into the computer bay 290, elastic forces position contact bands 620 against the grounding plate of an inserted ACM and establish a conductive path to ground potential via the bands' connection to the metallic computer bay case 291 at

IPR2024-00027
CrowdStrike Exhibit 1002 Page 462

ground potential. Alternatively, if the computer bay case 291 is not at ground potential, a dedicated wire (not shown) can be attached from a ground potential source directly to contact bands 620.

Similarly, the computer bay 290 includes a cooling component comprising heat sink 610, opening 611, mounting pads 612, and springs 614. The cooling component provides a mating mechanism for the cooling pad 530 (shown in **Figure 5**) on an inserted ACM. Heat sink 610 is mounted to project into the interior of the computer bay cavity through opening 611 at the top 293 of computer bay 290. Elastic force from springs 614 positioned against mounting pads 612 hold heat sink 610 in firm physical and thermal contact with the cooling plate 530 (shown in **Figure 5**) of an inserted ACM. Through effective thermal coupling between the ACM's cooling plate and the computer bay's heat sink, heat generated during operation can be conducted away from the ACM, via heat sink 610, to the interior of the PCON where air circulated by a cooling fan can dissipate the heat. Movable heat sink 610 may also be thermally coupled to computer bay case or frame 291 to provide greater heat-sinking mass and surface area for heat dissipation.

The computer bay 290 includes a locking component comprising latching arm 631, locking tip 631, opening 632, pivots 633 and 634, spring 635, mounting tab 636, release arm 637, and plate 638. The locking component operates to prevent egress of an inserted ACM from the computer bay 290. Locking tip 631 of the latching arm 630 projects into the computer bay cavity 299 through opening 632 in the side of computer bay 290. The locking tip 631 engages one of the notches 540 (shown in **Figure 5**) formed into the case of an ACM when the ACM is inserted into the computer bay 290. The position of opening 632 and locking tip 631 thus corresponds to the position of a notch 540 (shown in **Figure 5**) in the exterior case 510 of an operatively inserted ACM. The latching arm is pivoted at pin 633 with elastic tension from spring 635 urging locking tip 631 toward the interior of the computer bay cavity 299. The angled, frontward edge of locking tip 631 facilitates displacement of the locking tip toward the exterior of the computer bay when the solid portion of the ACM case 510 (shown in **Figure 5**) to the rear of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 463

the notch 540 (shown in **Figure 5**) is moving past locking tip 631 as the ACM is inserted into the bay 290.

Release arm 637 is pivotally connected by pin 634 to latching arm 630. Rearward pressure on release arm 637, such as from a computer operator depressing a front panel button (not shown) attached to the front surface of plate 638, causes latching arm 630 to pivot about pin 633 in opposition to the force provided by spring 635. Sufficient rearward motion of release arm 637 causes locking tip 631 to disengage adequately from the cavity space 299 to permit removal of the ACM from the computer bay 290.

One skilled in the art recognizes that the motion of locking tip 631 could be restricted, controlled, or actuated by a variety mechanical or electrical means. For example, a mechanical lock could be coupled to locking tip 631 to prevent egress of the locking tip from the cavity 299 without a key. Similarly, an electrical actuator such as a solenoid, possibly under software control, could operate to disengage the locking tip 631 only after entry of a security password by the user. One skilled in the art recognizes that similar alternatives exist regarding many aspects of computer bay construction, and that these alternatives may be employed in varying embodiments without departing from the scope and spirit of the invention.

Figure 6 illustrates the internal component layout for one embodiment of an ACM. All components are contained within the confines of the ACM case 510, except for connectors 330a and 330b which extend from the rear of the ACM 100 to engage mating connectors (not shown) that will couple the ACM circuitry with the PCON circuitry. Main circuit board 610 provides electrical connections for circuitry within the ACM and mounting for many of its components 124, 122, 126, 152, 142 148, 320, and 330. The fabrication and use of such circuits boards is well known and understood in the art. Connector 622 is also mounted on main circuit board 610 and mates with mobile processor module 620. Mobile processor module 620 represents a form of packaging for a microprocessor and related components. The illustrated mobile processor module 620 is a self-contained unit that includes a microprocessor 112, CPU cache 114, and CPU bridge 146 operatively interconnected by the manufacturer. An example of one such module is the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 464

Pentium Processor with MMX Technology Mobile Module from Intel Corporation (order number 24 3515-001, September 1997). One skilled in the art recognizes that discrete microprocessor, cache, and bridge could have been employed and mounted directly to the main circuit board.

The mobile processor module 620 blocks the view, from the top, of the system BIOS 124. Similarly, hard disk drive 132 hides RAM memory 122, the high performance graphics processor 152, the host interface controller 320, and flash memory 126. Memory upgrade socket 630 remains exposed to facilitate installation of additional RAM memory 122. Power regulator 142, like the memory upgrade socket, enjoys a generous amount of overhead clearance to accommodate its vertical size. The area including IDE controller 148 also enjoys overhead clearance to facilitate a cable connection with the hard disk drive 132.

The functional interconnection and operation of components contained within the ACM and depicted in **Figure 6** has already been described in relation to **Figure 4** for like numbered items appearing therein.

Figure 7 is a block diagram of a peripheral console (PCON). A peripheral console couples with an ACM to form an operating personal computer system. The peripheral console (PCON) supplies an ACM with primary input, display, and power supply; the ACM supplies the core computing power and environment of the user. In the presently described embodiment the physical PCON package 200 contains the PCON functional components 201 and the PCON side of the ACM-to-PCON Interconnection 300. The PCON functional components 201 comprise primary display 210, a primary input 220, a primary power supply 230, interface and support 240, secondary mass storage 250, other devices 260, and expansion slots 270.

The PCON side of the ACM-to-PCON Interconnection 300 comprises a Peripheral Interface Controller (PIC) component 340, a PCON connector component 350, console-type component 342, and flash memory device 348. The PIC 340 and connector 350 components couple the PCON functional components 201 with the signals of an ACM-to-PCON interface bus 310 used to operatively connect an ACM with a PCON. The ACM-to-PCON interface bus 310 comprises

IPR2024-00027
CrowdStrike Exhibit 1002 Page 465

conveyance for electrical power 314 and signals for a peripheral bus 312, video 316, video port 317, and console-type 318.

The PCON side of the ACM-to-PCON Interconnection 300 operates as a counterpart to the ACM side described earlier in reference to **Figure 4**. Accordingly, the ACM-to-PCON interconnection 300 shown in **Figure 7** conveys certain signals in bit-serial format. The ACM-to-PCON Interconnection 300 also uses differential signaling to convey certain bit-serial data.

More specifically, in the presently described embodiment at least the peripheral bus 312 has a plurality of serial bit channels numbering fewer than the number of parallel bus lines in PCI bus 241 and operates at a clock speed higher than the clock speed at which any of the PCI bus lines operates. The peripheral bus 312 includes two sets of unidirectional serial bit channels which transmit data in opposite directions such that one set of bit channels conveys serial bits from the PIC while the other set conveys serial bits to the PIC. For each cycle of the PCI bus 241 clock, each bit channel of the peripheral bus 312 transmits a packet of multiple serial bits.

PIC 340 includes a number of functional subsections that are not shown individually in **Figure 4**. The PIC includes a bus controller to interface with PCI bus 241 and to manage transactions that occur therewith. The PIC also includes a translator coupled to the bus controller to (1) encode control signals from PCI bus 241 into control bits and to decode control bits conveyed by peripheral bus 312 into generated control signals for PCI bus 241, and (2) to manage the movement of data and address bit information to and from PCI bus 241. Additionally, PIC 340 includes a transmitter and a receiver, each coupled to the translator. The transmitter converts parallel bits representing data, address, and control signals into serial bits and transmits the serial bits to the peripheral bus 312 using differential signaling. The receiver receives serial bits from a differential signal of peripheral bus 312, and converts the serial bits representing data, address, and control signals into parallel bits.

Connector component 350 may be selected to mate directly with the connector component 330 of an ACM (shown in **Figure 4**). Alternatively,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 466

connector component 350 may be selected to mate with, for example, the connector on one end of a cable intervening between the PCON and an ACM in a particular embodiment, such as cable 364 shown in **Figure 5B**. The ACM-to-PCON interconnection herein described has the advantage of providing reliable signal conveyance across low cost cables.

Flash memory device 348 provides non-volatile storage. This storage may be accessible to devices in both the ACM and the PCON, including the host interface controller and the peripheral interface controller to which it is connected. As such, flash memory 348 may be used to store configuration and security data to facilitate an intelligent mating between an ACM and a PCON that needs no participation of the CPU.

The primary display component 210 of the PCON functional circuitry 201 of the presently described embodiment comprises integrated display panel 212 and video connector 213. Integrated display panel 212 is a color LCD display panel having a resolution of 640 horizontal by 480 vertical pixels. 640-by-480 resolution is popularly considered to be the minimum screen size to make practical use of the application software in widespread use today. One skilled in the art recognizes that the type and resolution of the display can vary greatly from embodiment to embodiment, depending on factors such as cost and intended application. Any display device may be used, without departing from the scope and spirit of the invention, that provides principal visual output to the computer user for operating system and application software executing in its customary and intended fashion using the CPU component (110 of **Figure 3**) of an ACM presently coupled to PCON 200.

Integrated display panel 212 is coupled to video signal bus 249 and displays a screen image in response to video signals presented on bus 249. Certain pins of connector 350 receive video output signals 316 of the ACM-to-PCON interface bus 310 from a mated connector that is coupled to an ACM. These certain pins of connector 350 couple to video signal bus 249 which conveys the video output signals 316 throughout the PCON 200 as needed. Video connector 213 is exposed at the exterior of PCON 200 and couples to video signal bus 249. Connector 213

IPR2024-00027
CrowdStrike Exhibit 1002 Page 467

permits easy attachment of an external display device that is compatible with the signals carried by bus 249, such as a CRT monitor (not shown). The external display device may be used in addition, or as an alternative, to integrated display panel 212.

The isolation of the relatively heavy and sizable primary display 210 from the core computing power and user environment contained within an ACM represents a further advantage of the present invention.

The primary input component 220 of the PCON functional circuitry 201 of the presently described embodiment comprises keyboard interface circuitry 222, keyboard connector 223, pointer interface circuitry 224, and pointer connector 225. Keyboard interface circuitry 222 and pointer interface circuitry 224 connect to ISA bus 245 and are thereby coupled to the CPU component (110 of **Figure 3**) of any ACM attached to PCON 200. Keyboard interface circuitry 222 interfaces a standard computer keyboard (not shown), attached at connector 223, to ISA bus 245. Pointer interface circuitry 222 interfaces a standard computer pointing device (not shown), such as a computer mouse attached at connector 225, to ISA bus 245. Computer keyboards, pointing devices, connectors 223, 225, keyboard interface circuitry 222, and pointer interface circuitry 224 are well known in the art. The isolation of the relatively heavy and sizable primary input devices 220 from the core computing power and user environment contained within an ACM represents a further advantage of the present invention.

The primary power supply component 230 of the PCON functional circuitry 201 of the presently described embodiment provides electrical energy for the sustained, normal operation of the PCON 200 and any ACM coupled to connector 350. The power supply may be of the switching variety well known in the art that receives electrical energy from an AC source 289, such as a wall outlet. Power supply 230 reduces the alternating current input voltage, to a number of distinct outputs of differing voltages and current capacities. The outputs of power supply 230 are applied to power bus 231. Power bus 231 distributes the power supply outputs to the other circuitry within the PCON 200. Bus 231 also connects to certain pins of connector 350 to provide the electrical power 314 for an ACM

IPR2024-00027
CrowdStrike Exhibit 1002 Page 468

conveyed by ACM-to-PCON interconnection 300. The isolation of the usually heavy power supply 230 from the core computing power and user environment contained within the ACM represents a further advantage of the present invention.

The interface and support component 240 of the PCON functional circuitry 201 of the presently described embodiment comprises peripheral bridge 246, diskette controller 242, IDE controller 248, and signal conveyance paths 241, 243, 244, 245, 247 and 249. Peripheral bridge 246 couples PCI peripheral bus 241 with peripheral busses of other formats such as ISA peripheral bus 245 and others 247. PCI and ISA peripheral busses are industry standards, well known and understood in the art. Other peripheral busses 247 may include, for example, a bus compliant with the universal serial bus (USB) industry standard. While other embodiments of a peripheral console 200 may include a single peripheral bus that is coupled to an attached ACM via ACM-to-PCON interconnection 300, such as PCI bus 241, this embodiment includes peripheral bridge 246 to establish additional busses 245, 247. The additional busses 245, 247 permit the use of the many low-cost and readily available components compatible with these bus specifications.

Diskette controller 242 interfaces a floppy disk drive 254 with the CPU component 110 of an attached ACM (shown in **Figure 4**) so that the CPU may control and use the diskette drive 254 hardware to store and retrieve data. Diskette controller 242 couples to the CPU via a connection to ISA bus 245. Diskette controller 242 connects to the diskette drive 254 via one of device cables 243.

Similarly, IDE controller 248 interfaces a hard disk drive 252 and a CD-ROM drive 256 with the CPU component 110 of an attached ACM (shown in **Figure 4**) so that the CPU may control and use the hard disk drive 252 and CD-ROM 256 hardware to store and retrieve data. IDE controller 248 couples to the CPU via connection to PCI peripheral bus 241. IDE controller 248 connects to each of hard disk drive 252 and CD-ROM drive 256 via one of device cables 243. Some embodiments of PCON 200 may take advantage of VLSI integrated circuits such as an 82371SB (PIIX4) integrated circuit from Intel Corporation. An 82371SB integrated circuit includes circuitry for both the peripheral bridge 246 and the IDE controller 248 in a single package.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 469

The secondary mass storage component 250 of the PCON functional circuitry 201 of the presently described embodiment comprises diskette drive 254, hard disk drive 252, and CD-ROM drive 256. Secondary mass storage 250 generally provides low-cost, non-volatile storage for data files which may include software program files. Data files stored on secondary mass storage 250 are not part of a computer user's core computing power and environment. Secondary mass storage 250 may be used to store, for example, seldom used software programs, software programs that are used only with companion hardware devices installed in the same peripheral console 200, or archival copies of data files that are maintained in primary mass storage 150 of an ACM (shown in **Figure 4**). Storage capacities for secondary mass storage 250 devices may vary from the 1.44 megabytes of the 3.5-inch high density diskette drive 254, to more than 10 gigabytes for a large format (5-inch) hard disk drive 252. Hard disk drive 252 employs fixed recording media, while diskette drive 254 and CD-ROM drive 256 employ removable media. Diskette drive 254 and hard disk drive 252 support both read and write operations (i.e., data stored on their recording media may be both recalled and modified) while CD-ROM drive 256 supports only read operations.

The other devices component 260 of the PCON functional circuitry 201 of the presently described embodiment comprises a video capture card. A video capture card accepts analog television signals, such as those complying with the NTSC standard used for television broadcast in the United States, and digitizes picture frames represented by the analog signal for processing by the computer. Video capture cards at present are considered a specialty, i.e., not ubiquitous, component of personal computer systems. Digitized picture information from video capture card 260 is carried via signal conveyance path 244 to the peripheral interface controller 340 which transforms it to the video port signals 317 of the ACM-to-PCON interconnection 300 for coupling to the advanced graphics processor 152 in an attached ACM (shown in **Figure 4**).

Video capture card 260 is merely representative of the many types of "other" devices that may be installed in a PCON to expand the capabilities of the personal computer. Sound cards and laboratory data acquisition cards are other

IPR2024-00027
CrowdStrike Exhibit 1002 Page 470

examples. Video capture card 260 is shown installed in one of expansion slots 270 for coupling to the interface and control circuitry 240 of the PCON. Any of other devices 260 could be coupled to the interface and control circuitry 240 of the PCON by different means, such as direct installation on the circuit board that includes the interface and control circuitry 240; e.g., a motherboard.

The expansion slots component 270 of the PCON functional circuitry 201 of the presently described embodiment comprises PCI connectors 271 and ISA connectors 272. A circuit card may be inserted into one of the connectors 271, 272 in order to be operatively coupled with the CPU 110 of an attached ACM (shown in **Figure 4**). Each of connectors 271 electrically connects to PCI bus 241, and may receive and hold a printed circuit card which it electrically couples to PCI bus 241. Each of connectors 272 electrically connects to ISA bus 245, and may receive and hold a printed circuit card which it electrically couples to ISA bus 245. The PCI 241 and ISA 245 busses couple to the CPU 110 of an attached ACM (shown in **Figure 4**) by circuitry already described.

Figure 8 depicts internal major component placement for one tower desktop peripheral console (PCON). The components shown belong to desktop tower PCON 200 and are housed in PCON case 202, except for ACM 100. Motherboard 810 is mounted parallel to the right side of PCON case 202. This orientation is employed to comport with industry standard form factors for motherboards and desktop computer cases. Motherboard 810 provides mounting and interconnection for much of the circuitry of the PCON. For example, for the PCON embodiment illustrated in **Figure 7**, PCON-side interconnection circuitry 300, interface and control circuitry 240, keyboard 222 and pointer 224 interface circuitry, and expansion slot 270 components, could all be installed on motherboard 810.

Computer bay 290 is mounted with its front opening exposed at the front face of PCON case 202 allowing easy insertion of ACM 100. The long edge of the front opening of computer bay 292 runs perpendicular to the plane of motherboard 810. This orientation is employed to comport with industry standard form factors for desktop computer cases. Cable 364 carries signals between the ACM 100 and motherboard 810 containing PCON-side interconnection circuitry (not specifically

IPR2024-00027
CrowdStrike Exhibit 1002 Page 471

shown). Signals flow between cable 364 and ACM 100 over connectors 330, 362 mated when the ACM is inserted into the computer bay.

Because the rear face of computer bay 290 which holds connectors 362 runs perpendicular to the plane of motherboard 810, the use of a flexible cable simplifies the interconnection between them. Use of the ACM-to-PCON Interface already described facilitates the use of flexible cable.

Figure 8 also depicts CD-ROM 256, power supply 230, PCI expansion slot 271, and an "other device" 260, occupying positions that generally comport with form factors norms at use in the personal computer industry. Other Device 260 comprises a PCI expansion card plugged into PCI expansion slot 271.

Figure 9 depicts internal component placement for one notebook peripheral console (PCON) such as the one illustrated in **Figure 2b**. **Figure 9** shows a view from the top, with the hinged panel containing a display 212 opened as shown in **Figure 2b**, and with the keyboard and other top surfaces of the base portion 902 removed to allow a view of the interior. Motherboard 910 sits low within the base portion 902 of the notebook PCON 200. The motherboard lies in a plane parallel to the bottom surface of the base portion 902. Motherboard 910 provides mounting and interconnection for circuitry elements such as Peripheral Interface Controller 340, keyboard connector 223, pointer connector 225, and LCD display panel connector 920, and many others that are not shown.

Computer bay 290, battery 230, CD-ROM 256, and PCMCIA slots 930 are placed above motherboard 910. Computer bay 290 has its "front" opening 292 exposed at the right side of the notebook PCON case 202 to permit side insertion of an ACM into the computer bay 292. CD-ROM drive 256 has its front face exposed at the front face of notebook PCON case 202 to permit easy insertion and removal of CD media. Battery 230 which serves as the primary power supply for the notebook PCON has one face exposed at the left side of notebook PCON case 202 to facilitate insertion and removal of the battery 230. PCMCIA slots 930 are exposed at the right side of PCON case 202 to facilitate insertion and removal of PCMCIA cards into and from the slots 930.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 472

PCMCIA cards are credit-card sized electronics modules that extend the functionality of a personal computer system. PCMCIA cards and slots are well known and understood in the art. A PCMCIA card may, for example, contain a modem or network interface electronics.

The keyboard (not shown) of notebook PCON 200 is placed in a plane generally parallel to Motherboard 910, toward the front edge of notebook PCON 200, and above CD-ROM 256 and computer bay 290.

Various modifications to the preferred embodiment can be made without departing from the spirit and scope of the invention. (A limited number of modifications have already been described in the preceding discussion.) For example, a particular embodiment may insert another layer of bus bridging between the CPU bridge and the Peripheral bridge. This may be desirable if, for example, a vendor wants to implement a proprietary, general-purpose bus having intermediate performance characteristics that fall between those of the high-performance general purpose bus originating at the CPU, and the slower general purpose PCI bus. Thus, the foregoing description is not intended to limit the invention as set forth.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 473

CLAIMS

What is claimed:

1. A detachable computing module for attachment to a peripheral console for forming a fully operational personal computer system comprising:

an enclosure;

a CPU;

memory coupled to said CPU;

mass storage coupled to said CPU;

interconnection circuitry coupled to said CPU, said interconnection circuitry connectable to a peripheral console;

wherein said CPU is uncoupled from any primary input circuitry when said interconnection circuitry is disconnected from a peripheral console.

2. The apparatus of Claim 1 further comprising clock circuitry having a variable frequency output signal coupled to said CPU.

3. The apparatus of Claim 2 wherein said clock circuitry varies the frequency of said output signal in response to a signal presented at an input coupled to said interconnection circuitry.

4. The apparatus of Claim 3 wherein said signal presented at an input coupled to said interconnection circuitry conveys at least one functional characteristic of a peripheral console.

5. The apparatus of Claim 1 further comprising a cooling plate exposed to the exterior of said computing module and thermally coupled to said CPU.

6. The apparatus of Claim 1 further comprising a grounding plate exposed to the exterior of said computing module and electrically coupled to an electrical ground connection of the CPU.

7. The apparatus of Claim 1 wherein said enclosure further comprises projections forming a notch for engaging a latching mechanism.

8. A detachable computing module for attachment to a peripheral console for forming a fully operational personal computer system comprising:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 474

an enclosure;

a CPU;

memory coupled to said CPU;

mass storage coupled to said CPU;

interconnection circuitry coupled to said CPU, said interconnection circuitry connectable to a peripheral console;

wherein said CPU is uncoupled from any primary output circuitry when said interconnection circuitry is disconnected from a peripheral console.

9. The apparatus of Claim 8 further comprising clock circuitry having a variable frequency output signal coupled to said CPU.

10. The apparatus of Claim 9 wherein said clock circuitry varies the frequency of said output signal in response to a signal presented at an input coupled to said interconnection circuitry.

11. The apparatus of Claim 10 wherein said signal presented at an input coupled to said interconnection circuitry conveys at least one functional characteristic of a peripheral console.

12. The apparatus of Claim 8 further comprising a cooling plate exposed to the exterior of said computing module and thermally coupled to said CPU.

13. The apparatus of Claim 8 further comprising a grounding plate exposed to the exterior of said computing module and electrically coupled to an electrical ground connection of the CPU.

14. The apparatus of Claim 8 wherein said enclosure further comprises projections forming a notch for engaging a latching mechanism.

15. A detachable computing module for attachment to a peripheral console for forming a fully operational personal computer system comprising:

an enclosure;

a CPU having a power supply connection point;

memory coupled to said CPU;

mass storage coupled to said CPU;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 475

interconnection circuitry coupled to said CPU, said interconnection circuitry connectable to a peripheral console;

wherein said power supply connection point of said CPU is uncoupled from any electrical power source having sufficient energy to sustain execution of instructions by said CPU whenever said interconnection circuitry is disconnected from a peripheral console.

16. The apparatus of Claim 15 further comprising clock circuitry having a variable frequency output signal coupled to said CPU.

17. The apparatus of Claim 16 wherein said clock circuitry varies the frequency of said output signal in response to a signal presented at an input coupled to said interconnection circuitry.

18. The apparatus of Claim 17 wherein said signal presented at an input coupled to said interconnection circuitry conveys at least one functional characteristic of a peripheral console.

19. The apparatus of Claim 15 further comprising a cooling plate exposed to the exterior of said computing module and thermally coupled to said CPU.

20. The apparatus of Claim 15 further comprising a grounding plate exposed to the exterior of said computing module and electrically coupled to an electrical ground connection of the CPU.

21. The apparatus of Claim 15 wherein said enclosure further comprises projections forming a notch for engaging a latching mechanism.

22. A detachable computing module for attachment to a peripheral console for forming a fully operational personal computer system comprising:

an enclosure;

a CPU;

memory coupled to said CPU;

mass storage coupled to said CPU;

interconnection circuitry coupled to said CPU, said interconnection circuitry connectable to a peripheral console; and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 476

power supply circuitry having a stored energy capacity no greater than the energy required to power said CPU, memory, and mass storage for 30 minutes of operation at the maximum rated speed of the CPU.

23. The apparatus of Claim 22 further comprising clock circuitry having a variable frequency output signal coupled to said CPU.

24. The apparatus of Claim 23 wherein said clock circuitry varies the frequency of said output signal in response to a signal presented at an input coupled to said interconnection circuitry.

25. The apparatus of Claim 24 wherein said signal presented at an input coupled to said interconnection circuitry conveys at least one functional characteristic of a peripheral console.

26. The apparatus of Claim 22 further comprising a cooling plate exposed to the exterior of said computing module and thermally coupled to said CPU.

27. The apparatus of Claim 22 further comprising a grounding plate exposed to the exterior of said computing module and electrically coupled to an electrical ground connection of the CPU.

28. The apparatus of Claim 22 wherein said enclosure further comprises projections forming a notch for engaging a latching mechanism.

29. A detachable computing module for attachment to a peripheral console for forming a fully operational personal computer system comprising:

an enclosure;

a CPU having a power supply connection point;

memory coupled to said CPU;

mass storage coupled to said CPU;

interconnection circuitry coupled to said CPU, said interconnection circuitry connectable to a peripheral console; and

wherein said power supply connection point of said CPU conducts more than 20 percent of the power required for operation of the CPU at its maximum rated clock speed for no more than 30 minutes for any duration wherein

IPR2024-00027
CrowdStrike Exhibit 1002 Page 477

said interconnection circuitry is continuously disconnected from a peripheral console.

30. The apparatus of Claim 29 further comprising clock circuitry having a variable frequency output signal coupled to said CPU.

31. The apparatus of Claim 30 wherein said clock circuitry varies the frequency of said output signal in response to a signal presented at an input coupled to said interconnection circuitry.

32. The apparatus of Claim 31 wherein said signal presented at an input coupled to said interconnection circuitry conveys at least one functional characteristic of a peripheral console.

33. The apparatus of Claim 29 further comprising a cooling plate exposed to the exterior of said computing module and thermally coupled to said CPU.

34. The apparatus of Claim 29 further comprising a grounding plate exposed to the exterior of said computing module and electrically coupled to an electrical ground connection of the CPU.

35. The apparatus of Claim 29 wherein said enclosure further comprises projections forming a notch for engaging a latching mechanism.

36. An apparatus as in one of Claims 1-35 wherein said interconnection circuitry conveys peripheral bus signals from multiple parallel paths as sequential data bits on a single signal path.

37. An apparatus as in one of Claims 1-35 wherein said interconnection circuitry conveys electrical power, and signals comprising video output and peripheral bus information.

38. An apparatus as in one of Claims 1-35 wherein said interconnection circuitry conveys electrical power, and signals comprising video output, video port, and peripheral bus information.

39. A peripheral console for attachment to a detachable computing module for forming a fully operational personal computer system comprising:

a power supply;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 478

interconnection circuitry coupled to said power supply, said interconnection circuitry connectable to a computing module and for transmitting and receiving parallel computer bus signals as serial bit data streams;

40. The apparatus of Claim 39 further comprising a computer bay for holding an attached computing module.

41. The apparatus of Claim 40 wherein said computer bay further comprises a latching arm for mechanically engaging an attached computing module occupying said computer bay.

42. The apparatus of Claim 40 wherein said computer bay further comprises a heat sink.

43. The apparatus of Claim 42 wherein said heat sink comprises a dislocatable surface for applying pressure against an attached computing module occupying said computer bay.

44. The apparatus of Claim 40 wherein said computer bay further comprises a grounding contact.

45. The apparatus of Claim 44 wherein said grounding contact comprises a dislocatable surface for apply pressure against an attached computing module occupying said computer bay.

46. A personal computer system wherein the core computing power and environment for a computer user can be readily separated and transported from the remaining computer system components comprising:

(a) a detachable computing module including

an enclosure,

a CPU,

memory coupled to said CPU,

mass storage coupled to said CPU, and

(b) a peripheral console comprising a power supply;

(c) interconnection circuitry coupled to said computing module and to said peripheral console for conveying electrical power and signals between said computing module and said peripheral console;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 479

wherein said CPU is uncoupled from any primary input circuitry when said computing module is disconnected from said peripheral console.

47. The apparatus of Claim 46 wherein said interconnection circuitry conveys computer bus signals from multiple parallel signal paths as sequential data bits on a single signal path.

48. A personal computer system wherein the core computing power and environment for a computer user can be readily separated and transported from the remaining computer system components comprising:

(a) a detachable computing module including

an enclosure,

a CPU,

memory coupled to said CPU,

mass storage coupled to said CPU, and

(b) a peripheral console comprising a power supply;

(c) interconnection circuitry coupled to said computing module and to said peripheral console for conveying electrical power and signals between said computing module and said peripheral console;

wherein said CPU is uncoupled from any primary output circuitry when said computing module is disconnected from said peripheral console.

49. The apparatus of Claim 48 wherein said interconnection circuitry conveys computer bus signals from multiple parallel signal paths as sequential data bits on a single signal path.

50. A personal computer system wherein the core computing power and environment for a computer user can be readily separated and transported from the remaining computer system components comprising:

(a) a detachable computing module including

an enclosure,

a CPU having a power supply connection point,

memory coupled to said CPU,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 480

mass storage coupled to said CPU, and

(b) a peripheral console comprising a power supply;

(c) interconnection circuitry coupled to said computing module and to said peripheral console for conveying electrical power and signals between said computing module and said peripheral console;

wherein said power supply connection point of said CPU is uncoupled from any electrical power source having sufficient energy to sustain execution of instructions by said CPU whenever said computing module is disconnected from said peripheral console.

51. The apparatus of Claim 50 wherein said interconnection circuitry conveys computer bus signals from multiple parallel signal paths as sequential data bits on a single signal path.

52. A personal computer system wherein the core computing power and environment for a computer user can be readily separated and transported from the remaining computer system components comprising:

(a) a detachable computing module including

an enclosure,

a CPU,

memory coupled to said CPU,

mass storage coupled to said CPU, and

power supply circuitry having a stored energy capacity no greater than the energy required to power said CPU, memory, and mass storage for 30 minutes of operation at the maximum rated speed of the CPU;

(b) a peripheral console comprising a power supply; and

(c) interconnection circuitry coupled to said computing module and to said peripheral console for conveying electrical power and signals between said computing module and said peripheral console.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 481

53. The apparatus of Claim 52 wherein said interconnection circuitry conveys computer bus signals from multiple parallel signal paths as sequential data bits on a single signal path.

54. A personal computer system wherein the core computing power and environment for a computer user can be readily separated and transported from the remaining computer system components comprising:

(a) a detachable computing module including

an enclosure,

a CPU having a power supply connection point,

memory coupled to said CPU, and

mass storage coupled to said CPU;

(b) a peripheral console comprising a power supply;

(c) interconnection circuitry coupled to said computing module and to said peripheral console for conveying electrical power and signals between said computing module and said peripheral console;

wherein said power supply connection point of said CPU conducts more than 20 percent of the power required for operation of the CPU at its maximum rated clock speed for no more than 30 minutes for any duration wherein said interconnection circuitry is continuously disconnected from a peripheral console.

55. The apparatus of Claim 54 wherein said interconnection circuitry conveys computer bus signals from multiple parallel signal paths as sequential data bits on a single signal path.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 482

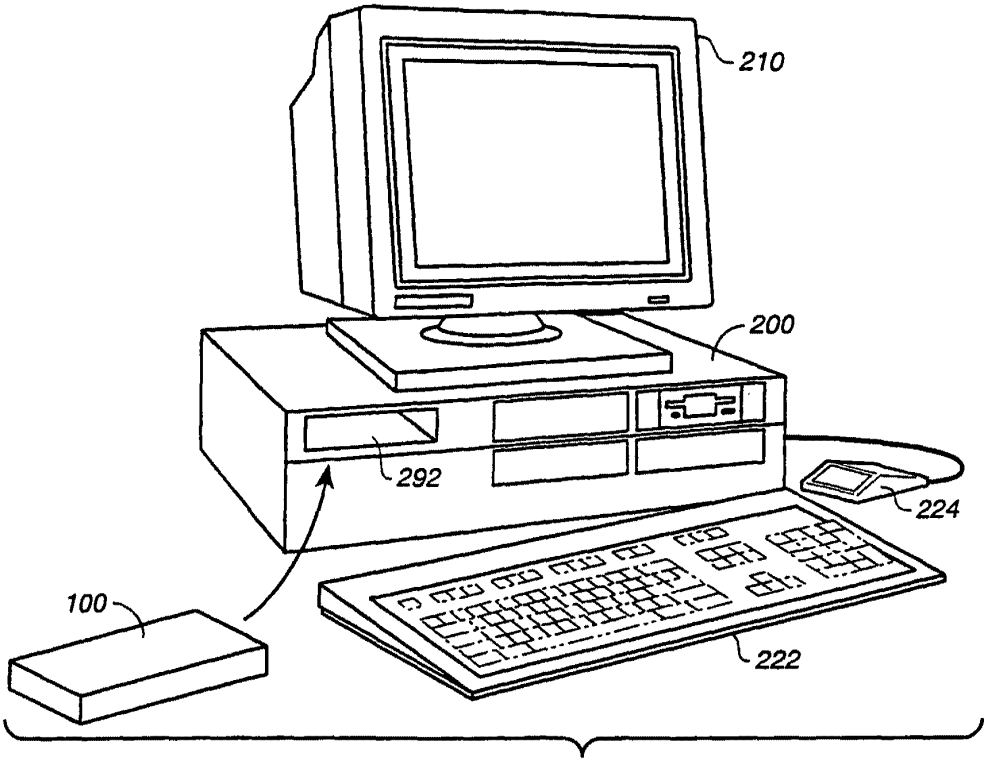


FIG._1

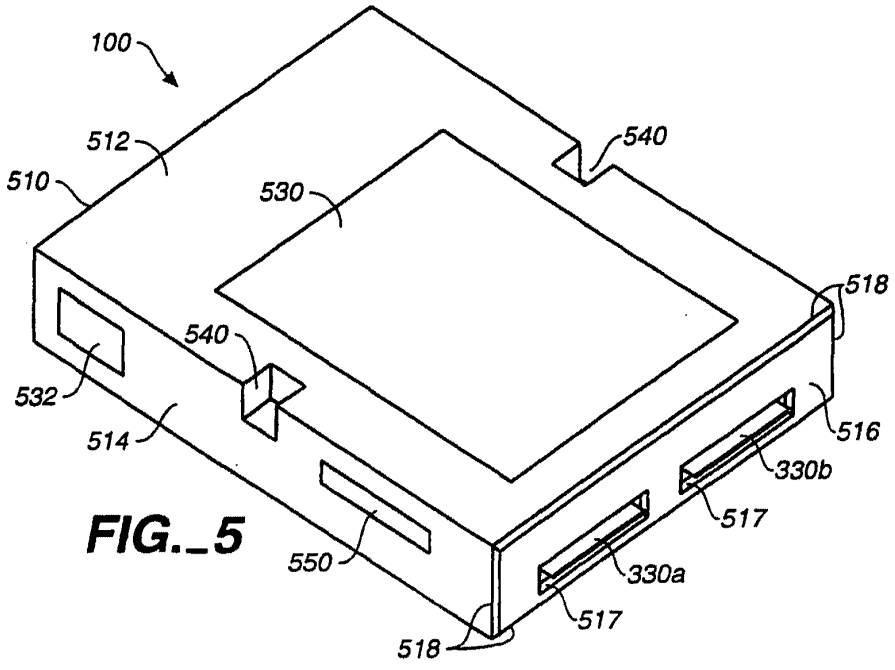


FIG._5

IPR2024-00027
CrowdStrike Exhibit 1002 Page 483

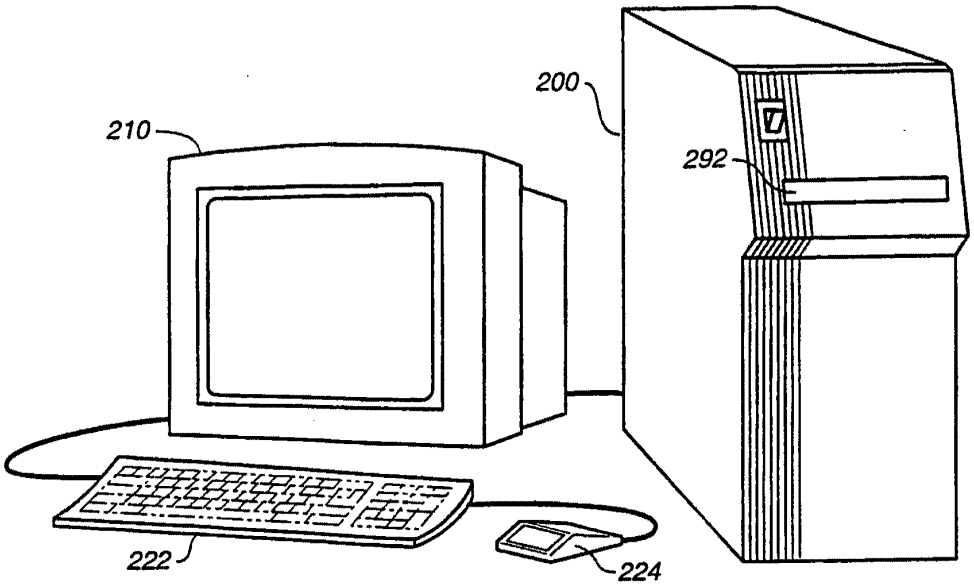


FIG._2A

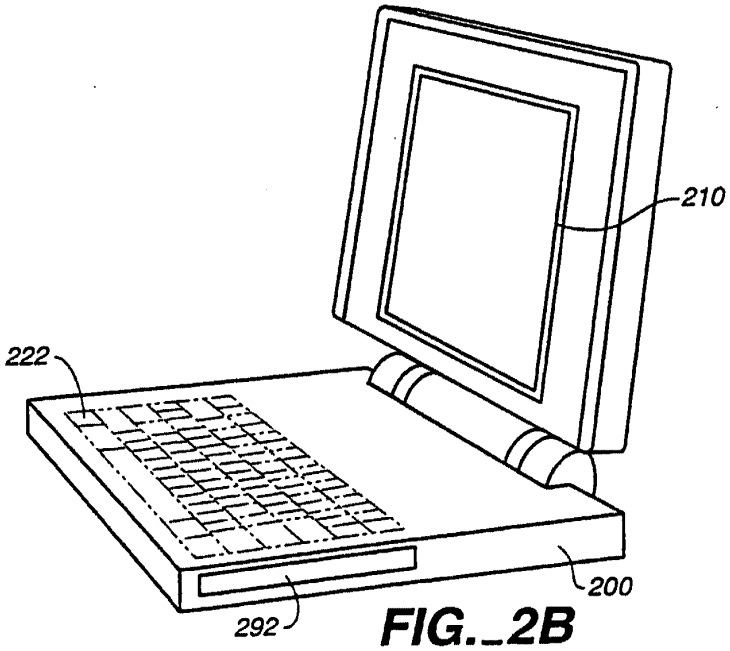


FIG._2B

IPR2024-00027
CrowdStrike Exhibit 1002 Page 484

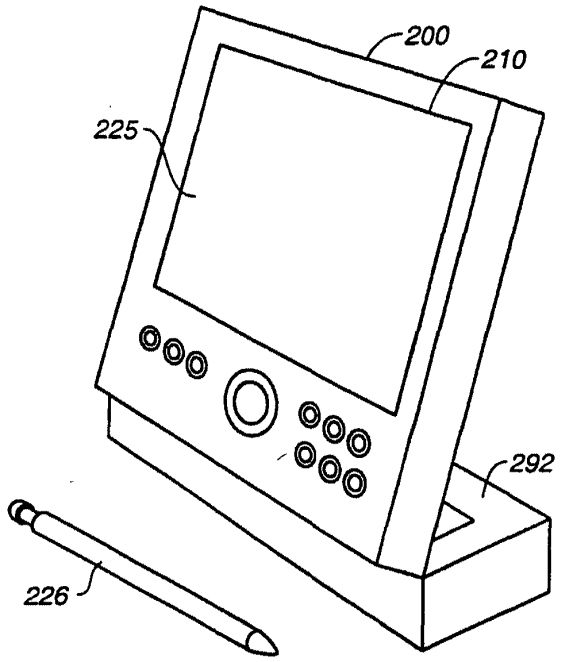


FIG._2C

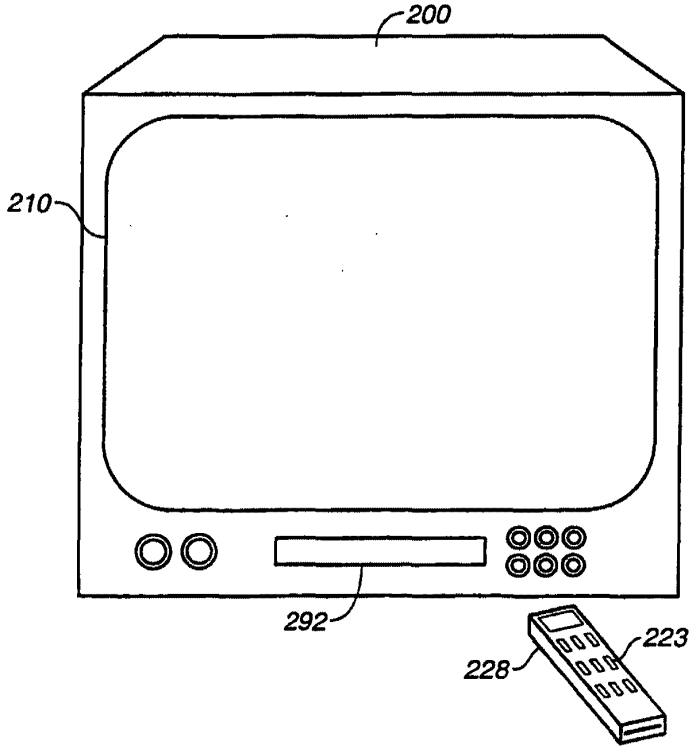


FIG._2D

IPR2024-00027
CrowdStrike Exhibit 1002 Page 485

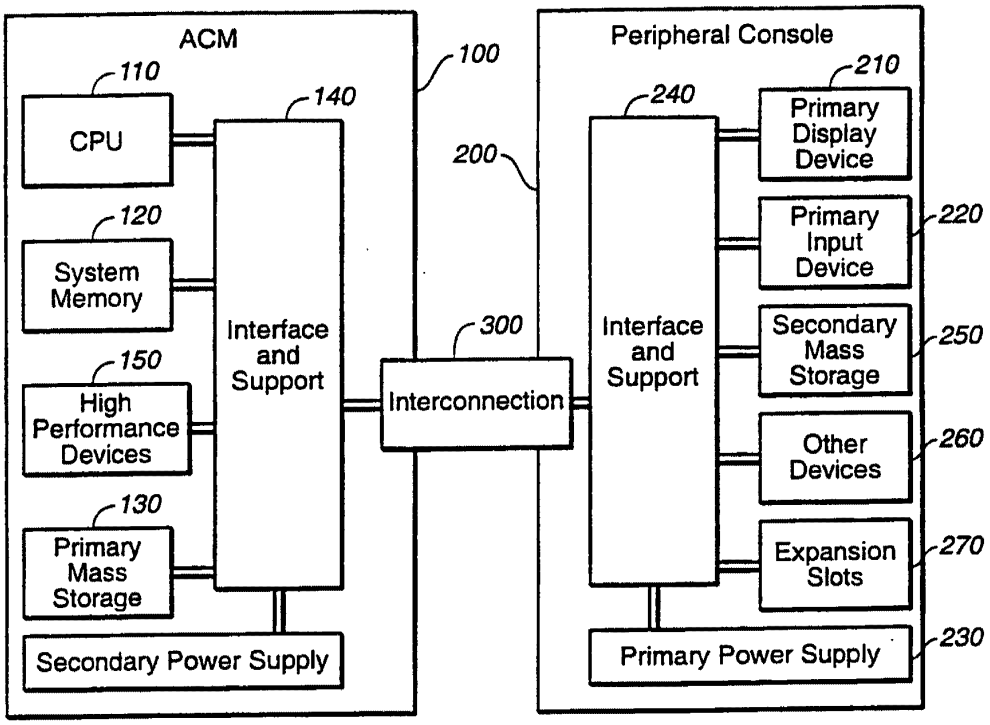


FIG._3

IPR2024-00027
CrowdStrike Exhibit 1002 Page 486

SUBSTITUTE SHEET (RULE 26)

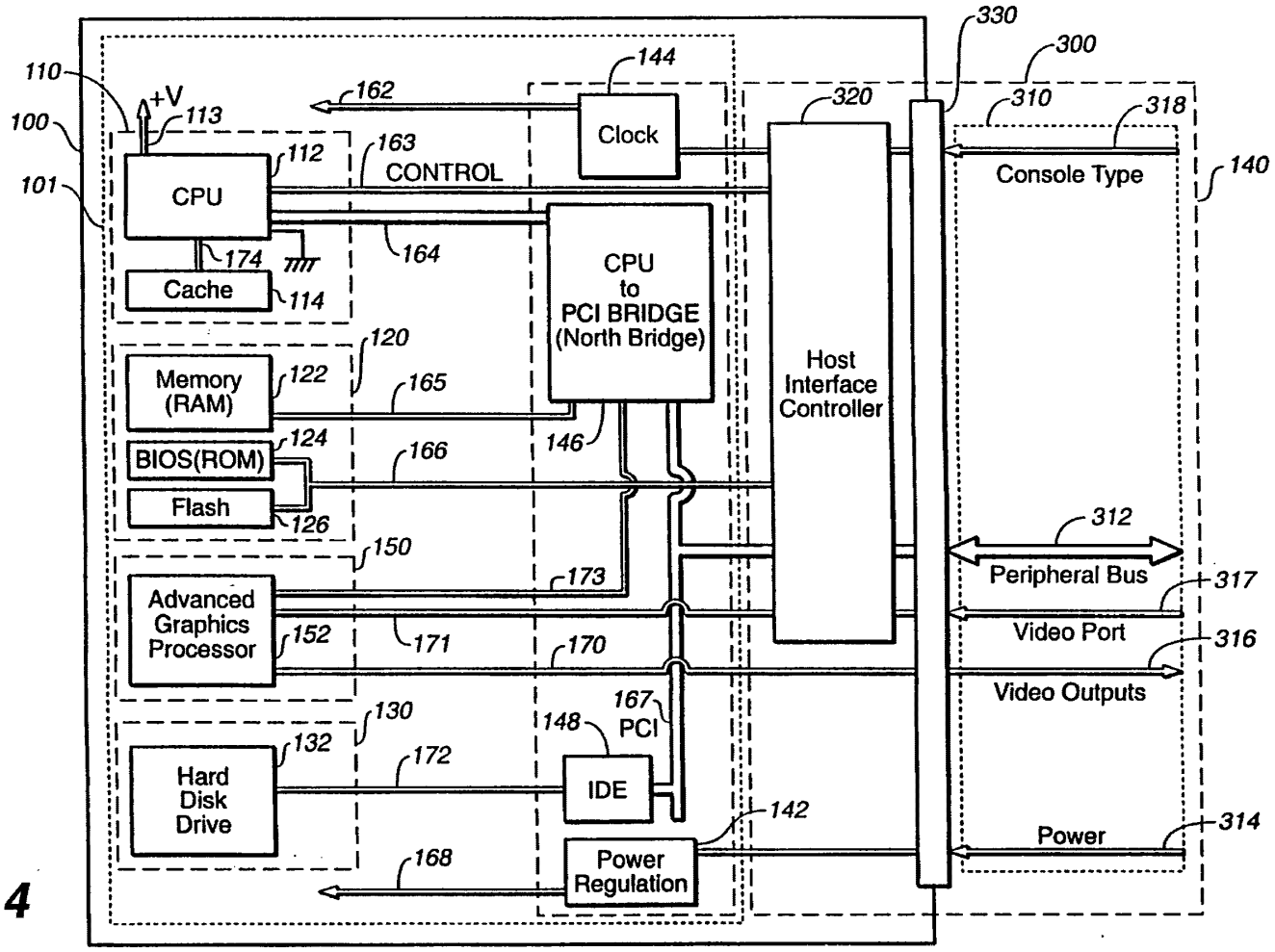


FIG._4

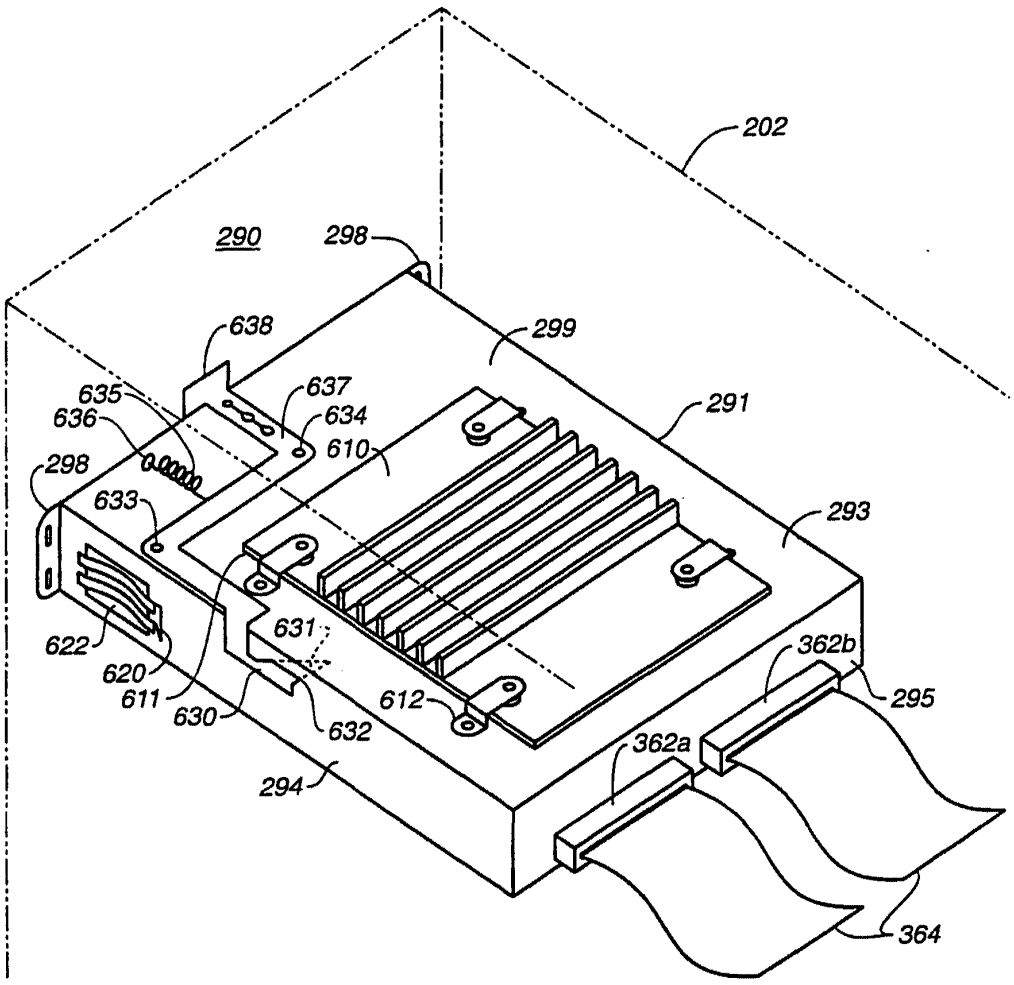

202
290
298
638
299
635
637
636
634
610
291
298
633
293
622
631
362b
620
611
630
632
612
295
362a
294
364

FIG._5B

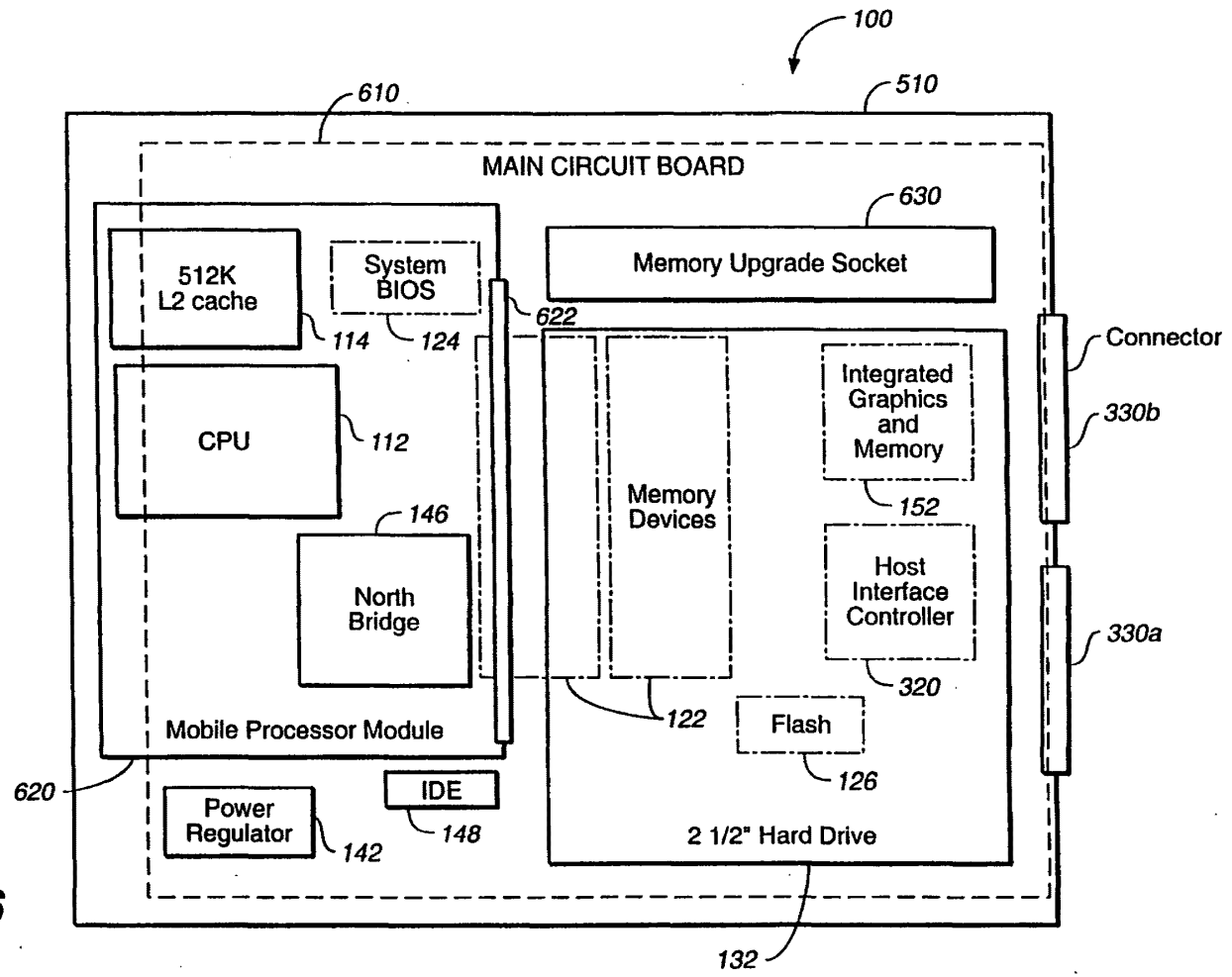

100
610
510
MAIN CIRCUIT BOARD
630
512K
L2 cache
System
BIOS
Memory Upgrade Socket
622
114
124
Connector
330b
CPU
112
Integrated
Graphics
and
Memory
152
Memory
Devices
146
North
Bridge
Host
Interface
Controller
320
330a
122
Flash
126
Mobile Processor Module
620
IDE
Power
Regulator
148
142
2 1/2" Hard Drive
132

FIG._6

SUBSTITUTE SHEET (RULE 26)

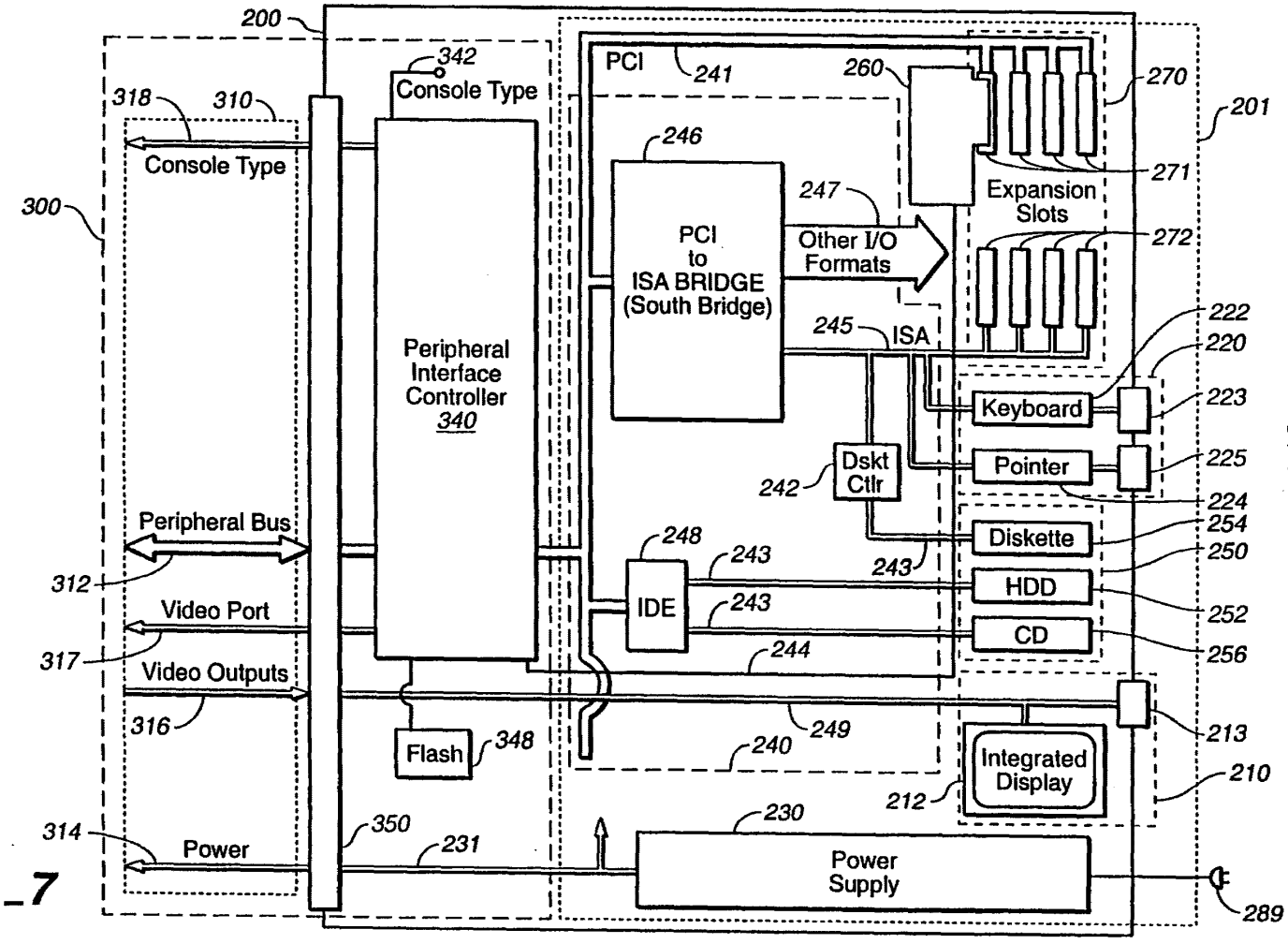


FIG._7

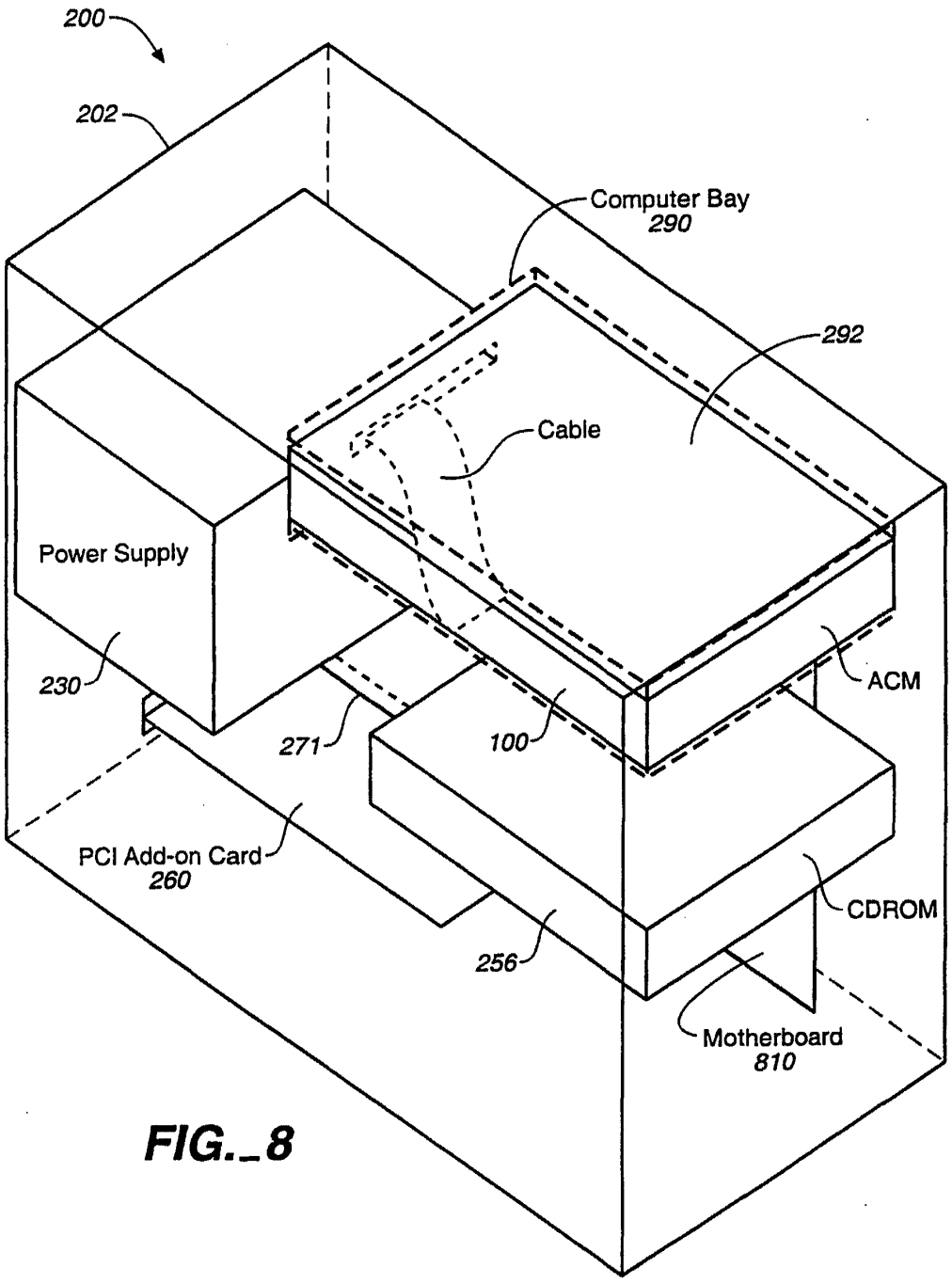


FIG._8

IPR2024-00027
CrowdStrike Exhibit 1002 Page 491

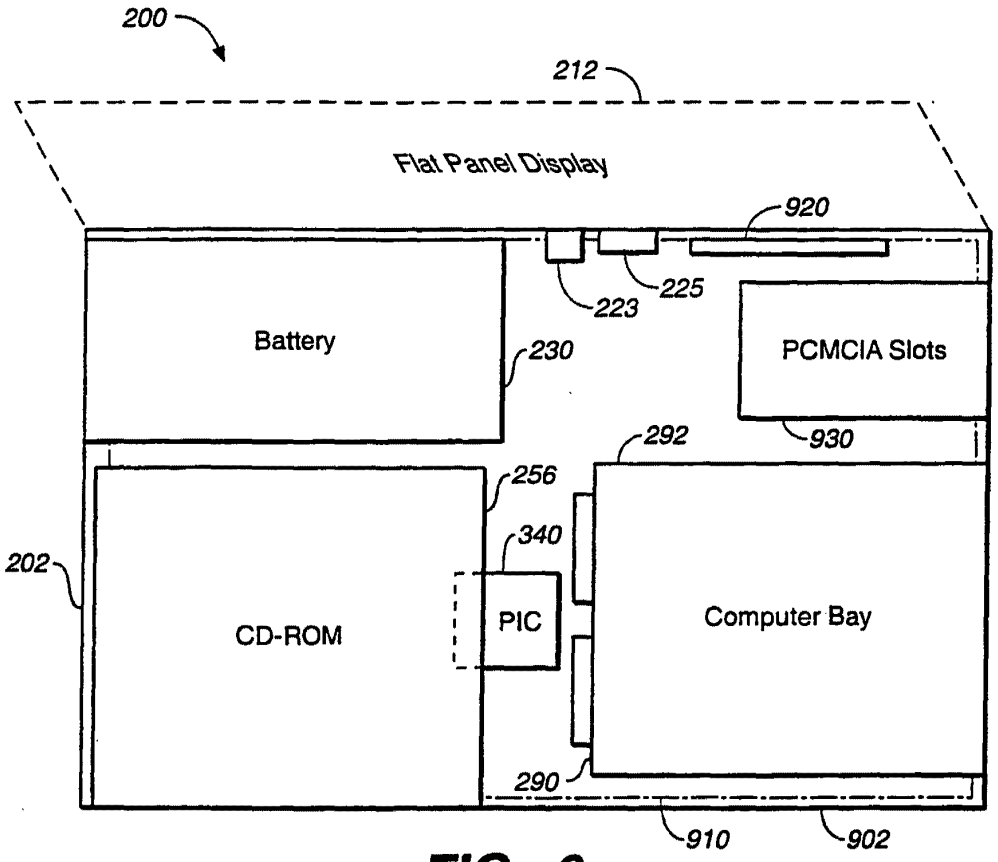


FIG._9

IPR2024-00027
CrowdStrike Exhibit 1002 Page 492

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04L — A2

(11) International Publication Number: **WO 00/10278**

(43) International Publication Date: 24 February 2000 (24.02.00)

(21) International Application Number: PCT/US99/16416

(22) International Filing Date: 21 July 1999 (21.07.99)

(30) Priority Data:
09/129,879 6 August 1998 (06.08.98) US

(71) Applicant *(for all designated States except US)*: CRYPTEK SECURE COMMUNICATIONS, LLC [US/US]; 14130-C Sullyfield Circle, Chantilly, VA 20151-1615 (US).

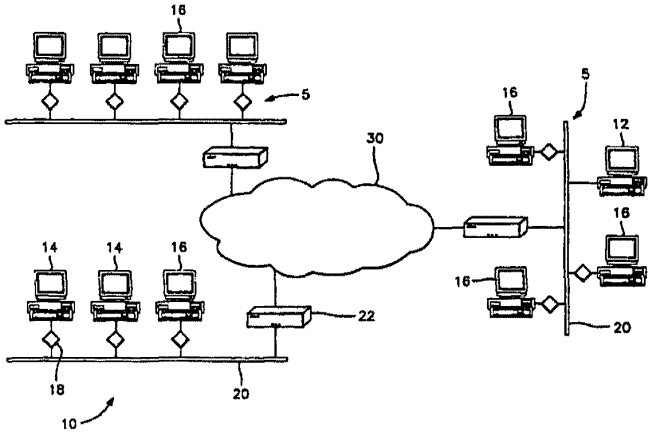
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: WILLIAMS, Timothy, C. [US/US]; 15122 Bernadette Court, Chantilly, VA 20151 (US).

(74) Agents: SLOBASKY, Michael, R. et al.; Jacobson, Price, Holman & Stern PLLC, 400 Seventh Street, N.W., Washington, DC 20004 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: MULTI-LEVEL SECURITY NETWORK SYSTEM



(57) Abstract

A network prevents unauthorized users from gaining access to confidential information. The network has various workstations and servers connected by a common medium and through a router to the Internet. The network has two major components, a Network Security Center (NSC) and security network interface cards or devices. The NSC is an administrative workstation through which the network security officer manages the network as a whole as well as the individual security devices. The security devices are interposed between each of workstation, including the NSC, and the common medium and operate at a network layer (layer 3) of the protocol hierarchy. The network allows trusted users to access outside information, including the Internet, while stopping outside attackers at their point of entry. At the same time, the network limits an unauthorized insider to information defined in their particular security profile. The user may select which virtual network to access at any given time. The result is trusted access to multiple secure Virtual Private Networks (VPN), all from a single desktop machine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 493

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Pinland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

MULTI-LEVEL SECURITY NETWORK SYSTEM

Field of the Invention

The present invention relates to a multi-level security network system. More particularly, the present invention relates to a secure communication between hosts using a network that implements a security policy, and especially a network allowing multiple levels of information to coexist on a network system.

Background of the Invention

The National Security Agency (NSA) has set forth specific definitions and requirements that establish various levels of security in computer and network systems. These basic definitions are set forth in "Trusted Computer System Evaluation Criteria," Department of Defense, 1985 (TCSEC) and "Trusted Network Interpretation of the Trusted computer System Evaluation Criteria," National Computer Security Center, 1987 (TNI). These documents define the requirements for systems to be evaluated in one of six hierarchical ratings: C1, C2, B1, B2, B3, and A1, with C1 being the least secure and A1 the most secure.

Division B, that is, ratings B1, B2, and B3, introduces the requirements for multi-level secure (MLS) systems. The term "multi-level security" refers to a system in which two or more classification levels of information are processed simultaneously, and not all users are cleared for all levels of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 495

information present. This same concept was applied during the 1980's to networked systems, at which time the phrase MLS network was generally used to refer to a network that was capable of simultaneously transmitting data at multiple security levels when some hosts or users were not cleared for all levels of data.

In order for an MLS network to qualify as a B-level secure network, it must provide at least the following five security functions: (1) access control, (2) object reuse, (3) labeling, (4) identification and authentication (I&A), and (5) auditing. Open Systems Interconnection (OSI) sets forth industry standard definition of seven layers of network connection: (1) physical, (2) data link, (3) network, (4) transport, (5) session, (6) presentation, and (7) application.

The first criteria, "access control," is concerned with the enforcement of rules (security policy) by which active subjects (e.g., processes, hosts) access passive objects (for example, files, memory, and devices). In a network system, operating at the OSI network layer of the protocol hierarchy, access control is concerned with the access of hosts to network packets. Rule-based Mandatory Access Control (MAC) is concerned with preventing each host from transmitting or receiving data at the wrong level. Discretionary Access Control (DAC), on the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 496

other hand, is concerned with ensuring that a host computer can only establish authorized connections to other hosts.

The second criteria, "object reuse" is concerned with preventing inadvertent release of residual data, typically in unused fields or at the end of a packet buffer. "Labeling" of each packet is necessary in a distributed system to convey the sensitivity of data to the various elements of the network. "Identification and Authentication" (I&A) is concerned with establishing individual accountability for authorized users. "Audit" is concerned with recording information about the use of the network's security mechanisms, to further support the requirement of user accountability.

In addition to these five basic requirements, a secure network should also provide two other capabilities, communications secrecy and communications integrity. These additional requirements support the secure transfer of MLS labeling and control information in an open environment, such as the Internet. Communications secrecy is provided by appropriate use of encryption to transform user data and control information so that it is unintelligible to wiretappers. Encryption is a process that scrambles or transforms sensitive data within messages (either an entire message, or part of a message) to make

IPR2024-00027
CrowdStrike Exhibit 1002 Page 497

them unreadable to any recipient who does not know a secret string of characters, called a key.

Communications integrity, on the other hand, is concerned with detecting modification of data, such as security labels, and user data, as it traverses the network. Packet integrity has also been accomplished by calculating cryptographic checksums of packet headers and packet data. The receiving node can straightforwardly detect message modification to a high degree of probability by recalculating the cryptographic checksum on the received data, and comparing it to the received checksum.

The current approaches to MLS networking include Verdix VSLAN (which has subsequently changed to GKI, then to Cryptek Secure Communications VSLAN), Boeing MLS LAN and ITT networks.

The Verdix Secure Local Area Network (VSLAN) product was developed by Verdix Corporation in the 1980s. VSLAN was the first network product evaluated by the TNI criteria and the first commercial network product to provide MLS security. VSLAN is the only commercial network product available with a B2 rating. However, VSLAN operates at the link layer (layer 2) of the protocol stack and, thus, its security mechanisms are limited to the scope of a local area network. While VSLAN uses Data Encryption Standard (DES) for communications integrity, it cannot

IPR2024-00027
CrowdStrike Exhibit 1002 Page 498

be used on an open network because DES is not sufficiently strong to protect classified data.

The Boeing MLS LAN has received an A1 security rating with respect to the TNI. It does not provide any encryption, but relies on physical protection of the medium to protect data in transit.

The ITT network security is described in U.S. Patent No. 5,577,209 to Boyle et al. ("Boyle"). Boyle uses cryptographic sealing techniques to support MLS labeling and mediation. The approach operates at the session layer (layer 5) of the OSI protocol reference model. Boyle, however, does not provide encryption of data for purposes of secrecy. Consequently, classified data could be accessed by passive wiretapping or by use of readily available tools, such as tcpdump running on any host in any of the intermediate networks.

Most protocol architectures do not have a protocol implementation that uses a distinct session-layer protocol. Rather, for protocols commonly used on the Internet, application programs (layer 7) interface directly to the transport layer (layer 4) of the protocol hierarchy. For these two reasons, Boyle is not very useful with real classified data on real networks, such as the Internet.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 499

Over the past 15 years, computer security and network security have progressed on separate tracks. Computer security has generally been concerned with the evaluation of standalone computer systems, without networking, at a time when the Internet was exploding. With the exception of a few TNI-evaluated products, network security has concentrated on the use of cryptography (particularly public key cryptography) and firewalls. Cryptography has been used to provide secrecy and integrity, largely without regard for the security of the communicating systems.

Various methods and devices have been used to enhance network security, including firewalls, identification and authentication (I&A), intrusion detectors, and virtual private networks (VPN).

Firewalls have been used to protect an organization's internal resources from the external Internet by passing certain protocols (e.g., email, name services) into the protection perimeter, but filtering out all protocols not explicitly listed. The firewalls attempt to isolate a company's intranetwork from the remainder of the Internet. Firewalls provide proxy servers that stand between the external network and internal resources and that pre-validate external requests.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 500

However, firewalls are only intended to protect internal resources from outsiders. As a result, firewalls offer no protection against an internal attack upon those same resources. Firewalls are generally subject to impersonation, in which an intruder's host is programmed to use an IP address of one of the network computers, or an attacker may know the password of one of the trusted users. Firewalls are also subject to session stealing, in which the intruder takes over an authorized session, together with all access modes established for the authorized user.

Firewalls may provide an IP tunneling capability to provide encrypted channel across the Internet to another part of the company's intranetwork. However, a major problem with firewalls is that once an unauthorized user is "in" to a host of the internal network, it is possible to use that host as a point for attacking other hosts within that network.

I&A devices generally involve the use of passwords for a user on one host to gain access across a network to another host. I&A may also be implemented by the use of a Personal Identification Number (PIN) and device to authenticate the user. The I&A device may be hardware using smart card technology that either takes a challenge and calculates a response or uses some clock-based synchronization to ensure that the I&A data passed

IPR2024-00027
CrowdStrike Exhibit 1002 Page 501

across the network is unique from one access attempt to another. I&A may also be implemented in software.

However, the use of I&A mechanisms does not address mediation (MAC, DAC) in any way, nor does it address the use of encryption or packet integrity on a stream of packets from one host to another. Furthermore, the risks of reusing a password across a network are obvious, given the availability of public domain software to grab packets as they traverse internal networks. Once an attacker (whether inside or outside the network) is able to send packets to your host, there are other attacks, such as routing via your host to another host, or sending packets to unprotected TCP/UDP ports, that could jeopardize the operation of any host on the network.

An intrusion detector generally involves the use of a "sniffer," such as tcpdump or hardware that snoops packets on the internal network, that attempts to detect and react to malicious behavior. However, intrusion detectors only provide detection, and not prevention. That is, the detectors cannot prevent an intrusion, but only detect that an intrusion has occurred. Intrusion detectors also do not provide a way for mediating the flow of packets, protecting packet secrecy, or protecting packet integrity.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 502

Virtual private networks (VPNs) are implemented by firewalls with encryption between different sites of a network or local area network (LAN). Some VPNs provide software-based encryption that runs on the individual host computers of the network. Generally, a VPN may be defined as a private network that exists within a larger, open network and uses methods (such as encryption) to make the network private.

Site-Level VPNs have the classic problems of firewalls. That is, they do not protect internal communications, and are susceptible to session stealing and sniffing on local and remote networks. An encrypted path between two sites may shut out an external attacker, but an internal attacker on the LAN has free rein to attack any host in that network, as well as any host in any of the other related LAN sites.

Software-based host level VPNs provide a driver that sits between the ethernet driver and the TCP/IP protocol stack. These VPNs offer some of the distributed advantages of having a firewall-at-each-host architecture, but do not have an architectural basis for protecting the hosts. The software VPN can be bypassed, for instance, if an application process on the host can communicate directly with the NIC driver via the OS on that host, instead of using the software VPN interface. Software VPNs may also be disabled or modified by processes running with

IPR2024-00027
CrowdStrike Exhibit 1002 Page 503

privilege (e.g., the UNIX root user or its equivalent on NT boxes). It relies on the host OS for protection, and not all hosts provide equivalent protections, so a node with a weak OS could be used as a location to launch attacks against other hosts.

Summary of the Invention

Therefore, it is a primary object of the present invention to overcome the limitations of the conventional security mechanisms. It is a further object of the invention to provide a secure network in which the security mechanisms are at layer 3 of protocol hierarchy. It is another object of the invention to provide a centralized administration of a layer 3 secure network that may be distributed over the Internet. It is another object of the invention to provide a security device that prevents unauthorized third parties from gaining access to a host. It is another object of the invention to provide a multi-level secure network having a security device coupled between each host and the network medium.

In accordance with these and other objectives, the secure network of the present invention, having a commercial name DiamondTEK Ultra, provides a unique implementation of cryptographic technology with high-assurance policy-based

IPR2024-00027
CrowdStrike Exhibit 1002 Page 504

enforcement of packet flow. The network prevents unauthorized users from gaining access to sensitive information. Although the network of the present invention has not yet completed evaluation by the NSA, it is designed to satisfy all of the B2 criteria.

The network generally comprises various host computers (such as PCs, workstations, or servers) connected by a common medium. Routers may be used to connect Local Area Networks to public networks, such as the Internet. The network has two major components, a Network Security Center (NSC) and security network interface cards or "security devices."

The NSC is an administrative workstation through which a network security officer manages the security aspects of the system and implemented by the individual security devices. The NSC is responsible for setting up authentication materials and for authorizing individual peer-to-peer and client server associations. In addition, since this is a Multi-Level Secure network, the NSC is responsible for defining the security levels and categories of information permitted on the network and the labeling used by each host.

The security devices are interposed between each host, including the NSC, and the common medium. The security devices enable a host to communicate across the network. Unlike other network security solutions, the security device is designed as

IPR2024-00027
CrowdStrike Exhibit 1002 Page 505

a self-contained circuit board that is directly integrated into the hardware of the host system. The architecture of the security device is readily integrated into a variety of hardware environments.

The security devices authenticate principals responsible for host computers that connect to the network. Through use of the security devices and the NSC, the network allows trusted users to access outside information, including the Internet, while stopping outside attackers at their point of entry. At the same time, the network limits an insider to information defined in their particular security profile. The user may select which logical network to access at any given time. The result is trusted access to multiple secure Virtual Private Networks (VPN), all from a single desktop machine.

The present network simultaneously enforces two independent and complementary types of access control rules: discretionary access control (DAC) and mandatory access control (MAC).

The network further ensures confidentiality and integrity of host-to-host communications by the use of encryption mechanisms. Cryptography provides the underlying secrecy and integrity of communications required for the network to be able to enforce a unique policy when operating over an open backbone

IPR2024-00027
CrowdStrike Exhibit 1002 Page 506

network. Various types of cryptography are available, including DES, and Triple-DES for commercial and financial applications, and high-grade Type 1 encryption for classified applications. The network ensures that messages cannot be modified by an unauthorized user, or attacker, through the use of message digests.

The features of the network overcome the problems associated with traditional I&A devices, intrusion detectors, firewalls and VPNs, as well as with previous MLS networks (such as VSLAN, Boeing MLS LAN and the ITT network). The security device is a separate hardware board having a separate CPU, memory, network interface and bus architecture from the application processes on the host. A dual-ported RAM architecture guarantees that no malicious host process can tamper with the internal configuration of the security device. All network accesses must go through a security device, which implement security mechanisms for each and every access attempt. The security devices cannot be bypassed since there is no other path to or from the network.

The network extends the mediation and cyptographic protection offered by a firewall (with its mediation and its cryptographic protection) to the network interface of each individual host computer. This may still permit a malicious user

IPR2024-00027
CrowdStrike Exhibit 1002 Page 507

on a particular Bethesda machine to use a permitted association to a host in Tampa to attack that machine, but the network architecture significantly restricts the ability of the attacker to access other hosts. Further, the auditing by the network involved, transparently forwarded to a central collection site at the NSC, can be used to provide accountability, which makes discovery for such attacks more certain.

The network provides hardware based mediation (MAC and DAC) at each host, and provides cryptographic protection (secrecy, integrity) on all host-to-host associations.

As used herein, the term "policy" refers to the control of sensitive and potentially classified data according to the rules of the system as set by the system controller. For example, a host operating at a security level of Secret may transfer data to another Secret host, but is prohibited from sending data to a host operating at a lower classification. Similarly, hosts operating at higher classifications or with additional non-hierarchical categories are prohibited from sending messages to the Secret host.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 508

Brief Description of the Drawings

Figures 1-3 show various instances of the secure network having a security device in accordance with the present invention.

Figure 4 is a diagram of the conceptual network architecture.

Figure 5 is a diagram of the packet format, including packet labeling, used on the network.

Figure 6 depicts representative labeling for the hierarchical levels and non-hierarchical categories used by the network.

Figure 7 is a representation of the transmit and receive windows for communication of information between hosts of the network.

Figure 8 is a block diagram of the security device.

Figure 9 is a flow diagram showing the IP packet transmission for the security device.

Figure 10 is a flow diagram showing the IP packet reception for the security device.

Figure 11 is a schematic showing conventional network adapter architecture.

Figure 12 is a schematic showing the host bust interface using a two-port RAM.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 509

Figure 13 is a block diagram showing the overall structure of the network.

Figure 14 is an example of how the security device can be used to partition a network (including the Internet) into multiple trusted Virtual Private Networks (VPNs), with the ability to switch a host between VPNs.

Detailed Description of the Preferred Embodiments

In describing a preferred embodiment of the invention illustrated in the drawings, specific terminology will be resorted to for the sake of clarity. However, the invention is not intended to be limited to the specific terms so selected, and it is to be understood that each specific term includes all technical equivalents which operate in a similar manner to accomplish a similar purpose.

Turning to the drawings, Figs. 1-3 show various embodiments of a security network 10 having a dedicated Network Security Controller (NSC) 12, workstations 14 and servers 16. The NSC 12 permits a Security Officer to configure and audit the operation of the secure network 10. The network 10 also has security devices 18, having the commercial name DiamondNIC, installed between each host (workstation 14 or server 16) and the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 510

local area network medium 20 to form a Local Area Network (LAN) 5. The various LANs 5 are connected to an untrusted backbone net 30 by a router 22.

The security device 18 is preferably a Network Interface Card (NIC) that easily replaces a standard NIC card for nodes that contain or need access to sensitive information. The security device is a self-contained circuit board that is directly integrated into the hardware of the host system.

Thus, the security cards 18 operate at the network layer (layer 3) of the protocol stack and provide encrypted, controlled communications from one host (IP address, TCP/UDP port) to another. Because the security mechanisms operate at layer 3, the security mechanisms pertain to the entire Internet address space. Further, the choice of LAN or WAN medium is not relevant to the security provided by the system.

Each security device 18 enforces a mandatory access control (MAC) policy, as well as discretionary access control (DAC) policy, on the flow of packets to and from that host 14, 16. It ensures labeling of all data packets with a hierarchical security level and a set of non-hierarchical security categories appropriate for the local host.

The DiamondNIC security device 18 provides user identification and authentication (I&A) via a card reader and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 511

keypad (not shown) attached to the device 18. Auditing is provided by the use of embedded firmware within the security device, with audit data directed to the NSC 12 for archival. Finally, the security device 18 uses encryption to provide secrecy and communications integrity on all selected connections.

Hosts that are installed with the DiamondNIC security devices 18 may communicate with like hosts on the internetwork, as permitted by profiles, operating at security levels loaded from the NSC 12. Host software, even malicious host software, cannot bypass the security mechanisms (mediation, auditing, encryption) provided by the security device 18 because the security mechanisms are embedded within the device 18 itself and all network communications must pass through the security device 18 in order to access the network.

The network 10 provides the same capabilities as firewalls, with several additional advantages. The network 10 extends the firewall concept to each individual host in the intranet. Thus, the policy enforced for each host is the policy required for that host, not a one-size-fits-all policy imposed by a single firewall. The network 10 also provides centralized network control, which permits hosts to be switched from one security profile to another, without risk that information may

IPR2024-00027
CrowdStrike Exhibit 1002 Page 512

leak across security levels. This network security architecture makes an NSA B2 security rating possible.

NETWORK SECURITY THREAT MODEL

To be effective, security mechanisms should be derived from the security threats that affect an organization. All security threats generally are concerned with the unauthorized disclosure of sensitive information or the modification of data. The primary threats for an organization processing sensitive data within a networked system, and particularly a networked system interconnected with the Internet, are disclosure, downgrading, passive wiretapping, active wiretapping, downloaded software, and covert channels.

In a classified environment, the threat of disclosure includes the ability to read data that is classified above the user's current level. In a classified environment, the threat of downgrading includes writing data to a lower classification than the user's current level. These threats are addressed by appropriate use of multi-level security (MLS) technology.

Passive wiretapping includes monitoring at intermediate sites, using tools such as tcpdump, as well as attaching devices to monitor the communications medium. The network counters the threat of passive wiretapping by appropriate use of encryption.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 513

Active wiretapping, also known as message stream modification, includes the modification of selected data (e.g., monetary amounts) within a packet, insertion of new packets into the data stream, playback of packets, and deletion of selected packets. Network control, as well as user data connections, can be attacked. This is countered by appropriate use of cryptographic checksums.

Cryptographic checksums are used to calculate an error detection code on a block of data, using encryption and a secret key. If two communicating hosts each calculate the same code, then there is a very high probability that the data was not modified in transit.

The threat of downloaded software includes viruses, malicious programs, Java code, and other software that can be downloaded by a trusted host from potentially malicious hosts. This is addressed in the present network by ensuring that hosts interact only with other trusted hosts operating at the same classification.

A covert channel exists when a high-level process manipulates a shared resource or modulates the rate at which data is sent, to signal data to a lower-level process. There are two types of covert channels, timing and storage. Covert channels, however, are much less a problem in a network environment than

IPR2024-00027
CrowdStrike Exhibit 1002 Page 514

on a standalone computer, but still must be addressed in MLS systems. Secure networks cannot entirely stop covert channels between communicating hosts.

ARCHITECTURE

The network 10 architecture essentially comprises a specialized NSC 12 host dedicated to configuring and auditing the secure network and a DiamondNIC network security device 18 installed between each host computer and the network medium 20.

Hosts, Users, and Principal

The network over which the security device 18 communicates actually enforces security with respect to network hosts. A principal is an individual that authorizes one or more users to access the network from a given host system, subject to a certain security profile (mandatory and discretionary access control rules). The network can be configured by the network security officer such that a single host may have more than one principal. Hence, each principal must complete an identification and authentication (I&A) procedure before the host is permitted to communicate over the network.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 515

When the I&A procedure has been completed, the security device 18 communicates with the NSC and downloads the principal's operational profile -- the combination of association lists (for discretionary access control) and security windows (for mandatory access control) -- from the NSC. From that point onward, the security device securely transmits and receives data over the network independently and transparently, relying upon its own CPU to avoid depriving the host of processing bandwidth. The bandwidth may be needed to offload processing, such as encryption. More importantly, however, independent transmission by the security device also prevents the host software from being able to bypass the security mechanisms.

The security device will only send and receive messages if the communication has been specifically authorized in the operational profile assigned by the network security officer. Encryption keys are generated and exchanged as necessary. The VPN is a collection of potentially communicating hosts, such as A, B, C, D and E. Each individual pair has an association, and the virtual private network (VPN) is the collection of all possible associations (e.g., A-B, A-C, A-D, A-E, B-A, B-C, etc.). Each pair of communicating security devices may be said to establish a transparent VPN, whereby every message is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 516

automatically encrypted before transmission and decrypted after arrival at its intended destination.

The network security officer may empower each user to access a variety of hosts with different degrees of privilege. For instance, suppose that a certain user is authorized to access the network not only from his desktop PC, but also from a workstation housed within a physically secure laboratory protected by a cipher lock. The NSO can define two operational profiles for the user, thought of as different roles, to give that user more privileges when accessing the network from the physically secure workstation than when connected from the unprotected desktop PC.

In addition, a person may be a principal at different devices with different profiles (that is, security levels and associations) defined for each device. A principal can also be enabled to operate the security device in a bypass or non-secured mode and thereby network non-sensitive data with other hosts that are not individually equipped with a security device.

Conceptual Network Architecture

Fig. 4 depicts the architecture of the system with respect to the trust required in its constituent parts and the OSI layers in the protocol hierarchy where these functions

IPR2024-00027
CrowdStrike Exhibit 1002 Page 517

reside. Region A includes the physical layer communications links, link layer protocols, repeaters, bridges, and intermediate routing hosts. The hardware and software in this region need not be trusted or physically protected, because of end-to-end mechanisms implemented in Region B.

Region B consists of the trusted devices, where each security device 18 is represented by a diamond-shaped object. The dashed lines indicate that control and auditing of the security devices 18 is implemented wholly within Region B, by the NSC 12. The devices 18 operate within layer 3 of the protocol hierarchy and provide a cryptographic foundation that assures communications secrecy and communications integrity. Any suited cryptographic method may be used, including the Data Encryption Standard (DES) and Triple-DES for commercial and financial applications and ranging to high-grade Type 1 algorithms for government and military applications.

Because the security devices 18 provide communications secrecy appropriate for the information being carried on the network, there is no need to physically or procedurally protect the communications medium in Region A. Any information intercepted by a wiretapper or intermediate host will be unintelligible. In addition, the security devices 18 provide communications integrity mechanisms appropriate for the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 518

information being carried on the network. Communications integrity mechanisms include, but are not limited to, keyed message digests (MDS), secure host algorithm (SHA) and message authentication code. Thus, any attempts to modify the host data (IP data field or header) will be detected by the recipient security device 18.

Regions C and D include host computers 14 (either workstations 14 or servers 16), the TCP/IP protocol stack, application programs, and users. The Internet Protocol (IP) runs at layer 3, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) run at the transport layer (layer 4) and application protocol (e.g., Telnet, File Transfer Protocol) clients and servers run at layer 7 of the protocol hierarchy.

The distinction between Region C and Region D is that hosts in Region C are trusted MLS computers that are capable of simultaneously processing data at multiple security levels, while hosts in Region D are not capable of simultaneously processing data at multiple security levels. Hosts in Region D may be evaluated according to the U.S. Government Trusted Product Evaluation Program (TPEP) or Trusted Technology Assessment Program (TTAP) programs, the upcoming Common Criteria, or less rigorous programs such as Security Proof of Concept Keystone (SPOCK) or the International Computer Security Association

IPR2024-00027
CrowdStrike Exhibit 1002 Page 519

(ICSA). These hosts may be multi-user or single-user at a time computers, but are capable of operating only at one security level at a time.

Installation and Profiles

The network 10 is typically installed by first installing the NSC 12 in a secure location, readily available to the security officers. The security device 18 adapters are typically installed in the backplanes of the various host computers, and software drivers, associated with the operating system, are installed on the host computers.

After the network 10 is installed and configured, the Security Administrator defines information about the security devices, their principals, and attached hosts. The administrator adds the information for each of the security devices 18 to a database located at the NSC 12. In addition, the administrator programs an authentication card for the principal with the security device information, including the principal's one-time password. The administrator travels to each of the nodes, and reads in the card to install node-specific information onto the board of the security device.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 520

A principal initializes and uses a node by first going to the node and, using the appropriate I&A means, selects a profile and identifies and authenticates himself/herself to the network. For example, this may include inserting the principal's authentication card in the card reader attached to the security device 18.

The principal also selects a usage profile using the pushbuttons on the front of the card reader. The principal is able to select only among those profiles entered by the security administrator. There can be up to 100 profiles defined for each principal although, in practical use, most principals will have only a few profiles. The selected profile has associated with it a security window and permitted host-to-host associations.

The NSC 12 sends the appropriate security window and associations for the profile to the security device 18. Once the host initializes the device driver, the host transmits packets to the network. The security device 18 mediates each packet according to the security window and authorized associations, then encrypts the packet using the appropriate traffic key.

When the security device 18 detects an attempted security violation, it sends an audit event to the NSC. If authorized, the principal may switch to a different profile, with a different security window and different associations. This

IPR2024-00027
CrowdStrike Exhibit 1002 Page 521

permits a principal to easily change from one usage profile to another, as required, among the set of such profiles authorized for the principal.

However, only one profile (that is, security windows and associations) can be in effect at a given security device at any time. Thus, if a principal change profiles by selecting a different profile, the security device is flushed and all data is reloaded. The principal may or may not have to remove the authentication card. The clearing out of the security device is not known by the principal.

The principal shuts down the security device 18 by removing the card (or logging out) and perhaps powering down the system. Throughout this operation, the security device 18 will only transmit or receive packets in accordance with the established current profile. Hence, the principal's profile is preferably configured to prevent access to both classified and public outside information at the same time.

Accordingly, the network allows trusted users to access outside information, including the Internet, while stopping outside attackers at their point of entry. At the same time, the network limits an insider to information defined in their particular security profile. The network preserves the security effects compartmentalization, while making it easy for authorized

IPR2024-00027
CrowdStrike Exhibit 1002 Page 522

users to access the information they need. Unlike-static VPN solutions, the network lets the user decide which network to access at any given time.

Integrity and Assurance

The network system ensures both confidentiality and integrity of host-to-host communications by the use of encryption and integrity techniques. Encryption is a process that scrambles sensitive messages to make them unreadable to any recipient who does not know a secret string of characters, called a key.

If the network security officer has authorized two hosts to communicate at a given security level, via the use of principal profiles, the security devices 18 perform a key exchange protocol and generate unique keys known only to that pair of security devices 18. These keys are henceforth used to encrypt all communication between the attached hosts at the chosen security level until one of the hosts either shuts down or disconnects from the network or a defined limit of use has been exceeded.

By having the security device automatically encrypt all messages, security of communication across the network is ensured. The user need not remember complex pass phrases and complete an authentication protocol that could be spoofed by

IPR2024-00027
CrowdStrike Exhibit 1002 Page 523

Trojan horse software. In addition, by applying encryption at the network layer, rather than at the application layer, the network accomplishes all authentication, key generation, and key distribution functions transparently and effortlessly. An application programming interface may also be provided by the security device 18 to allow the user to encrypt individual files and directories, as well as messages.

The assignment of unique keys to each pair of hosts at each security level offers two advantages. First, the network preferably incorporates various LANs, such as Ethernet and Token Ring, as well as transmitting packets through the Internet, which potentially allows messages to be intercepted by hosts other than the intended recipient. Encryption guarantees that only authorized target hosts can retrieve the information.

Second, if a key should somehow be stolen unbeknownst to the principals that share the key, only a small subset of the total network traffic is compromised. It is noted, however, that the traffic keys are not known even to the principals. The keys are established by the two communicating security devices, and kept in memory inside the two security devices. So, there's no practical way for someone to steal a key. Yet, if someone is snooping packets on the network, and manages to guess the key,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 524

the key would only be valid for traffic from one host, to another host, at a particular level.

Additional security may be provided by intermediate hosts on the Internet, such as Internet Service Providers, that run readily available sniffing tools, such as the UNIX tcpdump program to view all packets matching certain filters defined by the wiretapper.

The network 10 further ensures that messages have not been modified by an attacker through the use of message digests, such as cryptographic checksums. As noted above, a message digest is a number that is calculated from the text of the message and is then transmitted along with the message. When the encrypted information is received, the digest calculation is performed anew (for the encrypted data) at the receiving host and compared to the received value of the digest. If the transmitted value and newly calculated value match, the receiving host can be confident that the message was delivered intact.

Cryptographic Protocol Overview

Fig. 5 shows the preferred protocol headers for host-to-host messages and for control messages. All packets have an Ethernet or Token Ring header, as appropriate, with the standard IPv4 (Internet Protocol version 4) header and an IP Security

IPR2024-00027
CrowdStrike Exhibit 1002 Page 525

(IPSec) header extension with an Common IP Security Option (CIPSO) label, as specified by RFCs 1825-1829. Different packet formats, as yet unspecified, will be used for the Type 1 model of the security device 18.

The headers (IP, IPSec, CIPSO label, and cryptographic headers) are in clear text while IP data (i.e., TCP or UDP headers and data) are encrypted. RFC 1851 describes the formatting of encrypted packets. Label integrity uses, but is not limited to, DSS, SHA or MD5. This provides protection of both data and control communications. If tunneling is configured for the association, the host's IP header is encapsulated in the encrypted payload. At the receive end, before the packet is decrypted, MAC processing is based on the CIPSO label and DAC processing is based on the sending host's IP address.

Traffic keys are determined by each pair of communicating security device 18 using Internet Engineering Task Force (IETF) key determination based on shared secret information (IKE) or some other recognized process. The NSC 12 can be used as a Certificate Authority. Traffic keys are derived separately for each security level and host.

Key life is determined centrally, based on elapsed time or number of bytes. The NSC 12 defines the key life when the security device 18 is initialized and the security device 18

IPR2024-00027
CrowdStrike Exhibit 1002 Page 526

initiates its own key change when the key life expires. The security devices 18 transfer traffic keys to the NSC (via key generation audit) for short-term archival and potential key recovery.

Individual host-to-host associations may be protected by an appropriate encryption algorithm, as determined by the administrator. All network control communications are protected by the highest level of encryption available to the system. The network 10 also permits a network administrator to designate certain associations as clear-text (unencrypted). This mode of operation permits a host to communicate with other cleartext hosts.

SECURITY POLICIES

This section describes the Mandatory Access Control (MAC) and Discretionary Access Control (DAC) policies enforced by the security devices 18. It also describes Labeling, Identification and Authentication (I&A), Audit, Object Reuse, and System Architecture as they relate to the policy of the network.

Mandatory Access Control (MAC) Policy

Mandatory access control is implemented through a security window calculation (Fig. 7) at both the sending and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 527

receiving hosts. Each message is assigned a level that reflects both its sensitivity (e.g., secret, proprietary) and its handling restrictions (e.g., not releasable to foreign nationals, not releasable to the engineering department). Sensitivity is the combination of a hierarchical level (TopSecret > Secret > Proprietary > SBU > Unclassified) and a non-hierarchical category. Categories may be of the form "X," where X is the name of some project "Corporate Merger" or operation "Desert Storm."

Similarly, for at least single level workstations, each network principal, or authorized user, is assigned a level that reflects the host's trustworthiness to read and properly handle sensitive messages.

Two rules govern the reception and transmission of messages by the host. First, only an authorized host may receive sensitive data over the network. Thus, for instance, a standard PC operating as a host without any added security mechanisms of its own may have a security device configured to operate at Level-Top Secret and Categories=(USNukes, SpySatellites). This configuration would be okay provided that the host is not permitted to transmit any of that onto the network or to receive anything different. Thus, the security window for that host, but not that principal, is closed down so that the host can only SEND

IPR2024-00027
CrowdStrike Exhibit 1002 Page 528

TS (USNukes, SpySatellites) and RECEIVE TS (USNukes, SpySatellites).

Second, data that is transmitted is properly labeled so that it will be properly handled by the transmitting host as well as the remote receiving host. If the remote host is also a PC, then its security windows must be closed down to a single point as in the preceding example. The security device ensures that its transmissions are securely labeled. The network security officer is able to determine the specific range of levels, or security window, at which each host is permitted to communicate.

If the remote host is a Mulit-Level Secure computer that is capable of simultaneously processing a range of Secret to Top Secret, then the securtity window can be opened up accordingly. In this sense, the host is then trusted not to mix up the Secret with the Top Secret.

This flexible design permits unsecured hosts to be quickly and easily added to the network 10. These unsecured hosts can freely inter-operate with other unsecured hosts without any restriction. However, secured network hosts will not inter-operate with these unsecured hosts unless the secured hosts are specifically authorized to network non-sensitive data, that is, via a defined security window.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 529

MAC is concerned with preventing each host from receiving data classified above the host's current level ("read up") and transmitting data classified below the hosts's current level ("write down"). The level of a host computer is expressed as a range of hierarchical clearance levels (e.g., Secret, Top Secret) and sets of non-hierarchical categories (e.g., Project1, Operation2000), as determined by the profiles of individuals that use the computer. Individual host computers may or may not be trusted to handle MLS data.

As shown in Fig. 6, the network 10 is capable of supporting up to 256 hierarchical security levels and at least 65,535 non-hierarchical categories. The security administrator assigns names (e.g., "Unclassified", "Secret") and a hierarchical relationship to the security levels that will be used in the system. Typically, only a few security levels are defined and the rest are unused.

The administrator also assigns names to the non-hierarchical categories, but as the name implies, these can be placed anywhere in the category space. All hosts must use the same labeling conventions on the network, but individual MLS hosts may have different values for a level or category as represented within the operating system.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 530

Referring back to Fig. 4, the network 10 architecture provides an innovative and flexible "security window" mechanism that is capable of supporting both trusted MLS hosts (Region C) and single-level hosts (Region D). For hosts in Region C, the security device 18 can be configured to permit packets labeled at multiple levels, with multiple different sets of categories. There are separate windows for a host's transmissions to the network and the host's receptions from the network. The security window can be closed down to a single authorized level and set of categories, or can be opened up to accommodate multi-level hosts.

As shown in Fig. 7, packets {a, b, c} pass through the origin security device 18 transmit window, while packets {d, e} are labeled below allowable limits (attempts to "write down") and are rejected and audited. For the packets that pass through the sending host's transmit window, the packets are labeled by the sending host and mediated by the sending security device 18. At the destination, only packets {b, c} pass through the receive window while packet {a} is labeled above allowable limits.

As further shown in Fig. 7, in order for data to flow from one host to another, it must satisfy the MAC restrictions enforced by both the sending and receiving security device 18. The diamond shapes in the figure are intended to reflect the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 531

lattice organization of a range of security levels and non-hierarchical categories.

More formally, the requirements to transmit data are stated in terms of the dominance relationships. The hierarchical classification in the packet's security label must be greater than or equal to the minimum allowable classification in the host's transmit window. In addition, the packet's security label must be less than or equal to the maximum allowable classification in the host's transmit window.

The non-hierarchical categories in the object's security label include all the categories defined for the lower bound of the host's transmit window and are included in the categories defined for the upper bound of the host's transmit window.

A similar dominance relationship is defined for the receive side of the association. The important point, though, is that the purpose of this is to prevent the sending host from sending data below its actual classification ("write down") and prevent the receiving host from reading data above its authorizations ("read up").

As an example, suppose the labeling space defined for the network includes four hierarchical levels: Unclassified, Confidential, Secret, Top_Secret and Ultra_Secret; and 26 non-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 532

hierarchical categories {A, B, C, D, ... X, Y, Z}. In addition, suppose a B2-evaluated host is operating at a range of levels Secret to Top_Secret, with category A required, and categories {D, Z) optional depending on the data processed by the operating system. The security device 18 might be configured to permit any of the labels of Table 1 to be transmitted and received.

Level	Categories
Secret	A
Secret	A, D
Secret	A, Z
Secret	A, D, Z
Top_Secret	A
Top_Secret	A, D
Top_Secret	A, Z
Top_Secret	A, D, Z

TABLE 1

Continuing with our example, the security device 18 may also be configured by the system administrator to reject and audit any of the representative labels shown in Table 2.

Level	Categories	Reason
Secret	A, B	Category B is not permitted
Top_Secret	A, D, M, Z	Category M is not permitted
Secret	none	Must have Category A
Top_Secret	D, Z	Must have Category A
Unclassified	A	Security Level too low
Ultra_Secret	A, D	Security Level too high

TABLE 2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 533

For hosts in Region D, policy dictates that each host must operate at a single level at a time (otherwise it would be a non-MLS host processing more than one level of information). For these single-level hosts, the security window can be collapsed to a single labeling and all packets sent by the host must match that labeling exactly. For simplicity, however, a different option is permitted below.

Labeling - Single Level Hosts

The MAC implementation described above calls for each host to create a label in each packet, then to pass the packet to the security device 18 where the label is mediated with respect to the security window for that node. For single-level hosts, either the security device 18 can simply stamp the administratively-defined label into the packet header or a simplified driver can read the appropriate information from the security device and label the packets before giving them to the security device.

For single-level hosts, stamping the label into the header is equivalent to mediating the host-defined label against a security window that contains a single acceptable labeling. No mediation is required for these single level hosts because the security device 18 stamps the only label that is acceptable for

IPR2024-00027
CrowdStrike Exhibit 1002 Page 534

that node. The two approaches (security window, labeling) are compatible because, in both cases, (1) only packets consistent with the security policy are transferred onto the network, and (2) each packet on the network is labeled at the correct level.

The label can be at any hierarchical level and non-hierarchical category set defined for the network. The labeling information is transferred to/from the NSC 12 to the security device 18 in CIPSO format using the same means used to transfer the security window for multi-level hosts.

This method for stamping a label into the packet header works only for single-level hosts, or for MLS hosts that are administratively constrained to a single level on the network. Hosts that operate in MLS mode on the network must provide a trusted labeling process with the label mediated by the security device 18.

Discretionary Access Control (DAC) Policy

Discretionary access control at layer 3 of the protocol stack is concerned with ensuring that a host computer can only have associations with authorized host computers. DAC is implemented through a pair of association lists, a "receive" association list and a "transmit" association list. Both the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 535

receive and transmit association lists are assigned by the network security officer.

Each network host, or authorized user permitted to access an authorized workstation, can only send messages to hosts that appear in its transmit association list. Likewise, a host is only permitted to receive messages from hosts that are named in its receive association list.

The network 10 enforces a centralized discretionary access control (DAC) policy based on hardware addresses, Internet Protocol (IP) addresses and Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) ports. This policy is specified by a network security administrator at the NSC 12 and downloaded to each security device 18 as part of the principal's security profile. The DAC policy is not settable by the individual hosts involved.

The IP portion of the DAC policy addresses the ability of any host in the 32-bit IP address space to send and receive from any other host. This policy is enforced at the transmit side of the network (based on the sending host's authorization to send to the destination address) and at the receive side of the network (based on the receiving host's authorization to receive from the destination address). The DAC policy is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 536

independent of the classification level and categories, but cannot override a MAC decision.

In actual use, this may be set up, for instance, as associations between pairs of hosts. For example, a first set of hosts {A, B} may communicate with each other, and a second set of hosts {C, D, E} may also be permitted to communicate with each other. However, communication is not permitted between any of the hosts of the first set with any of the hosts from the second set. These sets of hosts are essential VPNs.

The network 10 also provides rudimentary port filtering based on TCP and UDP ports, with the default being no port-based filtering. TCP and UDP each provide 16-bits of port space, with the ports used to identify specific endpoints (client or server process) on the sending and receiving hosts. Each TCP/UDP header has a source port address and a destination port address, where the source port address is associated with the sending process on the source host, and the destination port address is associated with the intended recipient process on the destination host.

The port filtering rules are also part of the DAC policy, specified by a network security administrator at the NSC 12, downloaded to security devices 18 as they come online, and enforced by the security device 18 for every packet. The port

IPR2024-00027
CrowdStrike Exhibit 1002 Page 537

filtering aspect of the policy serves to further restrict communications between pairs of hosts that are authorized to communicate. For example, workstation A may be permitted to access server B, but the security device 18 for server B might block packets to port 23 (to block use of Telnet) or port 514 (to block access to the UNIX syslog facility).

The port filter is preferably per association, so effectively Telnet from a specific host may be permitted, and denied for all other hosts. For instance, if the protocol type does not specify either TCP or UDP, then the packet is passed to IP address filtering. Packets that are rejected because of a host-to-host association or a blocked port may be audited by the NSC.

The network 10 preferably combines the features of both mandatory and discretionary access control. Accordingly, for instance, in order for host A to transmit a message to host B, three conditions must be met: (1) the network requires that principal A's transmit association list must include host B; (2) host B's receive association list must include host A; and, (3) the security level of the message must be included within both host A's and host B's security windows. Requirements (1) and (2) are based upon DAC, and requirement (3) is based upon MAC.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 538

Identification and Authentication (I&A)

I&A is performed at the NSC 12 for operators and administrators and is performed at the security device 18 for the principals that are authorized to operate a node of the secure network. Identification of security officers at the NSC is a conventional login with user ID and password.

The security device 18 boards support various types of I&A for principals, including but not limited to authentication cards (what the user has) and ID/password (what the user knows). The mechanism used by a particular security device 18 board is determined from configuration data read from the administrator's card. The mechanisms include: an authentication card, ID/Password, Fortezza and Authentication card and PIN.

An authentication card is the standard means for I&A, and requires each principal to insert an authentication card in the security device 18 card reader to use the network. Each principal's card is programmed at the NSC and preferably transferred to the principal in person. The card contains one-time password information that prevents spoofing by anyone other than an administrator. This information includes the principal ID and a random value that is updated once per login.

The security device 18 performs I&A via an ID and password entered by the principal at the attached host. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 539

password is compared with the value stored at the NSC. This method requires the local host to provide a trusted path mechanism by which the principal can reliably place the user ID and password in the security device 18. For single-user workstations, this may mean, for instance, that the user of the workstation enter the principal ID and password.

Fortezza involves the use of a Fortezza card in the host system to sign a defined value with the identity of the principal. It also requires a trusted path between the host's Fortezza software and the network driver.

The authentication card and PIN mechanism requires the principal to enter a PIN at the host in addition to inserting the authentication card.

Regardless of the means used for authentication, a principal's ability to use the network depends upon the Security Officer defining the principal in a database at the NSC, and providing the means of authentication (programmed card, password, etc.) to the principal.

Audit

Audit is the second key part of an accountability policy. MLS systems must record information about security-relevant events such as use of I&A mechanisms and attempts to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 540

send data outside of the host's security range or on unauthorized connections.

The network 10 provides selectable auditing of the following types of events: login and logout of security officers at the NSC; change of security databases at the NSC; I&A of principals; statistical events, providing detailed information about the individual packets transmitted and received; exception events, including attempts to violate the security window, send to or receive from an unauthorized association, etc.; TCP/UDP port filtering rejections; and, TCP opens and closes.

The NSC provides real-time alarms of attempted security violations. These are typically directed to a printer, and include the date and time, principal identifier, IP addresses, and protocol type and port number. The network immediately notifies the network security officer of any attempted violations. The network optionally disconnects the offending host from the network to avoid additional audit data from being generated.

Object Reuse

Object reuse (OR) is concerned with preventing inadvertent release of residual data, typically, in unused fields or at the end of a packet buffer. The TNI has two requirements

IPR2024-00027
CrowdStrike Exhibit 1002 Page 541

for object reuse: preventing access to the residual data itself, and preventing use of residual authorizations. Both object reuse requirements are addressed by the innovative use of specialized hardware on the security device 18 board hardware that sanitizes buffers before they are reused again by the system. This hardware may be thought of as a macro extension of the processor.

System Architecture Requirement

A secure network must have a system architecture that ensures the network functions as a reference monitor. In accordance with the TNI, which defines the Network Reference Monitor (NRM) concept, an NRM must be tamperproof, must always be invoked, and must be small enough to be subject to complete analysis and testing. This relates to the ability of a MLS network to accurately and completely perform the functions that it is supposed to provide.

This requirement is inherently addressed by placing the security enforcement mechanisms in their own dedicated domain on the security device 18 board, where they cannot be interfered with by the host's software. It is supported by the effective use of cryptography to provide communications secrecy and communications integrity for all host-to-host transfers and all control communications.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 542

DESIGN

The secure network described above consists of a specialized NSC 12 host dedicated to configuring and auditing the secure network and a network security device 18 installed between each host computer and the network medium. The design of the NSC 12 and the security device 18 mediation will now be discussed.

Network Security Center or Controller (NSC)

The NSC 12 is a dedicated machine used by the security officer to configure, operate, and audit the operation of the secure network. The NSC is necessary for the network to run, although the security devices 18 may also be configured to continue operating without the NSC.

The NSC is responsible for both authenticating principals when they connect to the network and for authorizing connections. When a principal initially signs onto the network over an authorized security device, the NSC is contacted to verify the authentication data and to initialize security-relevant parameters, including the security profile, the association profiles, and the seed, or keying material, from which the security device generates individual encryption keys.

Once the initialization is complete, the security device possesses sufficient intelligence and autonomy to manage

IPR2024-00027
CrowdStrike Exhibit 1002 Page 543

all subsequent communication with other security devices across the network by itself. However, the security device continues to automatically report security-relevant occurrences to the NSC where they are displayed as real-time alarms and added to the audit log where they can later be examined for evidence of potential security violations.

The network architecture preferably provides for up to two separate NSCs, a primary NSC and a hot backup NSC. In normal operation, the primary NSC manages all aspects of the network, and provides automatic updates of network databases to the backup. The address of the backup NSC is known to the various security device adapters (from information read from an installation card), but the backup NSC does not participate in network policy management.

If the primary NSC fails, each security device independently switches over to the hot backup and periodically checks the availability of the primary. The backup preferably allows network principals to authenticate themselves, download configuration data, and begin operating on the network. The backup also logs audit data and provides the ability to configure individual boards to operate in "emergency mode" and generate their own keying material.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 544

The NSC is preferably implemented on a commercial off-the-shelf Pentium-class machine, using Windows NT for screen management, printer management, keyboard/mouse input, threads dispatching, and object reuse. The NSC software is organized as a main application with multiple worker threads for network control, network I/O, audit, print management, and system monitoring.

The NSC does not use any NT networking code. Instead, the NSC uses a security device 18 board with special firmware (known as the security device 18-Prime) that manages the encryption/transmission of control messages to the various security device boards, and the reception/decryption of responses and audit data. The security device 18-Prime manages control keys based on the security device 18 addresses.

The NSC 12 provides at least two levels of Security Officers in accordance with the concept of least privilege: an administrator has access to all NSC commands (configuration, operation, and auditing), while an operator can only perform restricted control and monitoring functions. A security administrator can define additional site-specific roles that permit operators at a particular facility to have additional privileges beyond the pre-defined operator role.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 545

The NSC command set includes security officer, network control, network management, principal, and security device 18 functions. The Security officer commands provide the ability for a security officer to login, logout, acknowledge alarms, and modify the security officer database. Network control functions include starting, stopping, suspending, resuming, and auditing the network.

Network management functions include defining user profiles, including security windows and host-to-host associations. It also includes loading key files, setting date and time, and database functions such as archiving the audit file and backing up and restoring databases.

Principals are the individuals that are directly responsible for the operation of nodes of the secure network. They may be users of single-user workstations, or may be administrators of multi-user systems. The functions related to management of principals include defining, editing, listing, and removing the principal data records, and programming access cards for the principals to bring up a network node.

Security device 18 functions include editing security device 18 data records, shutting down, suspending, and resuming security device 18 operation, and refreshing encryption keys used by the security device 18. Creating an installation card used

IPR2024-00027
CrowdStrike Exhibit 1002 Page 546

to installation of the security device boards with their IP address and control keys.

Security Device 18 Architecture

In the preferred embodiment, the security device 18 is a self-contained circuit board that is directly attached into the hardware of the host system. The architecture is straightforward, enabling the device to be readily integrated into a variety of hardware environments. The device operates using its own independent processor 48, bus 46, program and data memory 54. These independent elements isolate the security device, including its algorithms, and insure that it operates within its own protected domain. Providing an independent processor also avoids stealing any memory cycles or processing bandwidth from the host in which it is installed. Since the device operates at a low OSI level, enforcing all security mechanisms within the hardware circuit board, it is tamperproof and cannot be compromised by software-based attacks.

As shown in Fig. 8, the security device 40 consists of a single-board adaptor installed on a Host Backplane Bus 42. Different implementations exist for various host bus architectures (for instance, PCI, Sbus, and ISA) and various network interfaces (such as Ethernet and Token Ring). Yet, each

IPR2024-00027
CrowdStrike Exhibit 1002 Page 547

security device 40 includes a host interface, attached card reader 62, processor block, local RAM 54, cipher block and network block.

The host interface includes the host adapter's bus interface logic (not shown) and a block of two-port RAM 44. Portions of the memory 44 can be mapped into either the host adapter memory space or the security device's 40 own internal memory, but not both at the same time. The host copies a packet into the two-port RAM 44 buffer and writes a memory location to cause the buffer to be mapped into the security device memory. After the buffer is transferred to the security device, it is invisible to the host processor until it is processed.

The processor block consists of a dedicated processor 48, non-volatile memory 52 (EPROM/flash) for program storage, and support logic. The processor block includes dedicated hardware logic burned into a ROM on the security device board, to move buffers, scrub buffers in support of object reuse requirements, and move data to and from the encryption chip.

The security device has its own internal system RAM 54, which is used for transitory storage of data packets, security windows, association lists, and the like. A portion of the internal RAM 54, identified as network coprocessor RAM 64, is used by the network block to send and receive packet buffering.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 548

The cipher block 58, which consists of encryption device(s), support circuitry, and dedicated memory two-port RAM 56 available only to the processor and encryption hardware. The network block is the network coprocessor 66, which includes logic to send and receive packets on the network 68. The network 68 is preferably a LAN.

A key architectural feature of this hardware design is that the network medium 68 is separated from the host bus 42. This separation of the two interfaces dictates that packets will move from one interface to the other only if moved by security device's 40 software 52. The only way a packet may move from host bus 42 to local bus 46, is for the CPU 48, running the firmware 52, to grab the packet from the two-port RAM 44.

In addition, the hardware design provides a separation of the security device's own processing environment from both the host and the network. The security device's program and internal buffers are invisible to the host because of the dual-ported RAM design. Further, except for control requests from the NSC 12, which are accepted only from the NSC and must be cryptographically verified, there is no interface by which another host on the network can retrieve data from the security device's internal buffers.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 549

Because all communications from one host to another must use the services provided by the security device in order to access the network, it is not possible for a host to inadvertently or maliciously bypass the security device security features. In a properly configured network, where there are no other electrical connections to the network, it is possible to make absolute statements that the host software (whether trusted to operate in MLS mode, or not) must operate in accordance with the centralized network security policy set up by the security policy defined by the security officer at the NSC. Further, any packets that are transmitted are cryptographically protected before being placed on the network.

The security device functions are implemented in firmware 52 installed on the security device board. During installation, the security device firmware reads an administrator installation card at the authentication interface unit 62 to get the board IP addresses (Node, NSC, default router) and cipher key. Subsequently, the security device downloads principal-specific and node-specific data, via the network interface 66, from the NSC and sends audit events to the network for archival. However, the security device operates independently of the attached host.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 550

The security device has four general phases of operation: configuration, initialization, key exchange, and secure communication. Configuration is performed by the network security officer at the NSC workstation. The NSO configures each security device to support one or more principals, where each principal may have up to about 100 profiles. Each profile has associated mandatory access controls (security windows) and discretionary access controls (association lists).

Initialization of the security device occurs when a principal authenticates, via the security device, to the NSC. The security device reads security profile selected by the principal and cryptographic seed keying material from the database resident on the NSC. Whenever the security device establishes initial contact with another host that is also equipped with a security device, key exchange is conducted to prepare for secure communications between the hosts.

After a user is authenticated and selects a valid profile, the security device firmware downloads the security window and association list for the principal from the NSC. The window (that is, the hierarchical security levels and non-hierarchical categories, are represented in CIPSO format.

After configuration, initialization, and key exchange have taken place, secure communication between pairs of hosts is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 551

automatically and transparently managed by the respective security devices. User programs executing atop a security device equipped host only require access to a standard built-in networking application program interface (API) such as WinSock or TLI. This interface to the security device disguises the fact that an intricate sequence of operations is being performed by the security device each time a packet is transmitted or received across the network.

The network security claims for the security device 18 depend upon the proper sequence as well as execution of the following tasks. This can be viewed as multiple layers of processing in which the transmit flow of packets involves processing the individual layers from the highest layer (host) to lowest layer (network).

Packets are transmitted from the left to the right, that is, from the host bus 42 to the network 68, and received right to left, from the network 68 to the host 42. Packet flow is controlled by the CPU 48 and its firmware programs 52. Transmission and reception of packets will be discussed in further detail below with reference to Figs. 9, 10 and 12.

The security device 40 is configured so that malicious programs on the host computer can not directly access any of the devices on the security device local bus 46. Thus, direct access

IPR2024-00027
CrowdStrike Exhibit 1002 Page 552

cannot be established to the local RAM 54, network coprocessor 66, encryption hardware 58, or authentication interface 60. This is accomplished by only permitting communications between the host bus 42 and the internal bus 46 via the two-port RAM 44. Thus, the host bus 42 address signals, data signals, read/write signals, and so forth, do not extend to the local bus 46. Instead, each bus 42, 46 essentially terminates at the two-port RAM interface 44. (Fig. 12).

The two-port RAM 44 is connected to the host bus 42 and provides complete separation of the host and internal busses. The two-port RAM 44 is implemented as standard RAM storage devices with two separate bus interfaces. This design provides for no pass-through or leakage from one bus to the other, except by a write on one bus to a memory cell followed by a read on the other bus.

In operation, a principal inserts a card in the authentication interface unit 62 and selects a profile using the keypad and display on that unit. The unit 62 is typically connected to the interface logic 60 by a ribbon cabling. The logic 60 enables information to be read/write to/from the card, as well as to read the keypad and write to a display.

Once a principal is authenticated, the host may transmit packets to the network. With cross reference to Fig.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 553

9, a host asserts a signal on the bus 42 to put a packet into the two-port RAM 44, step 100. The CPU 48, operating under control of firmware 52, reads the packet from the shared memory 44 by asserting signals on the local bus 46.

The CPU 48, using a transmit association list in the internal memory 54, performs DAC by determining if the destination IP address is in the transmit list, step 102. The transmit list was previously downloaded from the NSC via the network 68, based upon the principal authentication and profile selected. The CPU 48, using a transmit security window in the internal memory 54, performs MAC by determining if the security label is consistent with the transmit security window, step 104. As with the transmit list, the security window was also downloaded from the NSC as a result of the principal authentication and the profile selected..

The CPU 48 then puts the packet in the cipher two-port RAM 56, which is used for communications with the cipher unit 58. The cipher unit 58 uses pre-loaded keying material to perform the cryptographic transformation and place the result back into the two-port RAM 56. Unlike the host two-port RAM, the cipher two-port RAM 56 preferably does not have protection responsibility. Rather, the cipher two-port RAM 56 is used to facilitate communications between the CPU 48 and cipher unit 58, step 106.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 554

Next, the CPU 48 collects information from the packet that is to be protected by a message digest (cryptographic checksum), and places it in the cipher two-port RAM 56. The cipher unit 58 makes the cryptographic transformation and puts it back into the cipher two-port RAM, where the CPU takes it and puts it in the IP header, step 107.

The CPU 48 then puts the packet into a reserved area in the network coprocessor RAM 64. The network coprocessor 66 takes the packet from this RAM 64 and transmits it onto the network 68, steps 108, 110. RAM 64 is a portion of the internal RAM 54 that is dedicated by the firmware 52 for use by the network coprocessor.

Packet reception operates in the reverse manner, as discussed with relation to Fig. 10. After the packet is received from the network 68 by the network coprocessor 66 and placed in the network coprocessor RAM 64, the CPU 48 takes over and does reception DAC, step 152 and MAC, step 154 using data structures in internal RAM 54 that were downloaded from the NSC after principal authentication and profile selection.

The CPU 48 verifies the integrity of the packet, step 156, by performing the same message digest calculation done by the sending host in step 107. If the results match, then the packet was not modified en route. This involves putting data in

IPR2024-00027
CrowdStrike Exhibit 1002 Page 555

cipher two-port RAM 56 and the cipher unit 58 performing a transformation using keys loaded in the hardware. The CPU 48 decrypts the packet, step 158, using the same general approach flow of processing as for encryption, step 106, but with the cipher unit operating in decryption mode.

Fig. 9 shows the process for transmitting information. At step 100, the host requests and the security device 18 maps the packet to be transmitted. At this step, the host places a packet in the Interface Control Block (ICB) and notifies the security device. This maps the packet out of host memory and into board memory.

At the security device, the packet then undergoes DAC, step 102. Here, the security device verifies that the host has an authorized transmit association to the destination address in the host's IP header. If not, an audit is generated, step 112, and the processing flow is terminated, step 114.

If the destination address is in the transmit list, Mandatory Access Control is performed. At step 104, the security device verifies that the host-specified packet label (CIPSO) is consistent with the transmit security window, or (for single-level hosts) labels the packet with the host label. If not, an audit is generated, step 112, and the processing flow is terminated, step 114.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 556

At step 106, for packets satisfying both discretionary and mandatory access control, the packet is encrypted, using the encryption key for the destination IP address. At step 107, the cryptographic checksum B is computed and placed in to the packet. Proceeding to step 108, the link layer header is generated, and at step 110, the packet is transmitted.

Similarly, packet reception occurs in an order of processing, from the lowest layer to the highest. Turning to step 150 of Fig. 10, the security device receives the packet from the network. Discretionary Access Control is then performed by security device by verifying that the host has a receive association for the source IP address in the incoming packet's IP header, step 152. If not, an audit is generated, step 162, and the processing flow is terminated, step 164.

Mandatory Access Control is performed at step 154 by verifying that the packet label (CIPSO) is consistent with the receive security window. At step 154, the security device verifies the integrity of the received packet by calculating a message digest (cryptographic checksum) of the received data. If the computed value matches the value sent by the originating security device, then the packet was not modified. If not, an audit is generated, step 162, and the processing flow is terminated, step 164.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 557

For packets satisfying both discretionary and mandatory access control, the packet is decrypted, step 156, using traffic key for source IP address. The security device then maps the packet out of the board memory and into the host memory.

When configured at installation (by data on the administrator's card), the security device 18 provides the ability for the attached host computer to initiate switching from one authorized profile to another. Each profile has associated with it separate transmit and receive security windows for MAC, as well as separate transmit and receive association association lists for DAC.

The host enters the profile by using its trusted path to write the new profile identifier into the security device ICB. The security device validates that the host actually has the requested profile and, if so, then resets the security window and associations and sends a profile change notification to the NSC. If the security device is not configured to allow the host to initiate profile switching, then principal must select another profile via the card reader in order to switch profiles.

The security device also provides a way for the local host to place cleartext data in the ICB and receive the encrypted results. This uses the standard packet transmission code with a specified key.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 558

Host Bus Interface Using Two-Port RAM

In order to better understand the operation of the host bus interface, which uses two-port RAM 44, reference will first be made to the conventional network adapter architecture. A standard network adapter 310 is shown in Figure 11. The same general architecture is used for other types of adapters, e.g., SCSI controllers, video controllers, etc.

Typically the adapter is plugged into the host bus 302, which typically consists of address lines 304, data lines 306, and control lines 308. For example, on a 32-bit computer, there might be 32 address lines, 32 data lines, and several control lines (interrupt, I/O, etc.). In a standard network adapter, some portion of these lines extend directly into the adapter card 310. For example, the local bus 311 might consist of 8 address lines, 8 data lines, and a few control lines. These are wired directly or with minimal interface logic to the host bus 302.

If the adapter has local RAM 312 or local adapter firmware 314, these are directly accessible to software (typically a device driver) on the host computer. This means is commonly used for personal computers to execute extensions to the device driver that are resident in firmware on the adapter board. In the case of a network adapter, a network coprocessor sits on

IPR2024-00027
CrowdStrike Exhibit 1002 Page 559

the bus and sends receives packets from adapter RAM-312 or from host memory (not shown).

The problem with this architecture for a security device, however, is that the contents of memory on the adapter board depend on the trustworthiness of the host operating system. Any user process on the host 300 that can bypass these host controls can modify memory locations on the adapter board and cause the network adapter to send or receive anything.

Turning to Fig. 12, the host bus interface having a two-port RAM is shown in further detail. The security device 310 runs in its own protected domain, completely isolated from potentially malicious host software running on the host CPU 300. This is done via a two-port RAM interface 312 implemented on the security device 310, and providing only a limited means for the host software to interact with the security device.

In particular, it is not possible for host software to observe or modify data in the local RAM 334, firmware 336, or network coprocessor 338. This is because the only signal lines that extend from the host bus onto the security device are those address and data lines used to read and write data into the host port 314 of the two port RAM 312. There is no path onto the local bus 320.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 560

The operation of the two-port RAM interface for packet transmission will now be discussed. The host 300 device driver builds a packet for transmission and writes the packet into a portion of the two port RAM. However, the exact memory layout of the two-port RAM is irrelevant to the security provided by the system, and other suitable configurations may be used.

The host 300 device driver writes to a memory location in the two-port RAM that causes an interrupt 326 to be signaled to the CPU 332 of the security device.

The firmware 336 of the security device contains code to map the portion of two port RAM 312 out of host memory and into the security device's memory. When this occurs, the memory is no longer visible to the host (typically, a bus error occurs if it is accessed) and is visible to the security devices firmware. Thus, when the transmit buffer is mapped out of host memory, a new buffer is mapped into host memory. Therefore the host always has a buffer to which he has access. The device will not map the buffer out of host space until it can process the buffer.

Next, the firmware 336 accesses the data by memory reads using the local address 322 and data 324 lines to the two port RAM. The packet is moved from the two port RAM area into local RAM space via a hardware assisted block move mechanism.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 561

The firmware 336 running on the local CPU 332 then performs MAC, DAC, encryption and integrity functions, and may hand the packet to the network coprocessor 338 for transmission. All of this is invisible to software running on the host CPU 300.

The operation for packet reception is similar. The network coprocessor 338 receives a packet, places it in local RAM, and the firmware 336 on the local CPU performs MAC, DAC, decryption, and packet integrity functions. If the packet is valid for the host, the firmware 336 places the packet in the two port RAM via memory writes using the address 322 and data 324 lines. Again, this goes into an available portion of the two port RAM using conventions established by the security device and the host device driver.

When the packet is fully placed into this memory, the firmware maps the memory buffer into host memory (thus making it visible to the host device driver) and writes a memory location that causes the two port RAM interface circuitry to assert an interrupt signal 308 to the host CPU.

The key points of this interface are: (1) host signal lines do not extend into the security device board, and so the only means for the host to interact with the security device is via this two port RAM interface. (2) A particular buffer in the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 562

two port RAM is invisible to the host while the security device is processing its contents.

The CPU of the security device has access to all memory on the NIC board at all times, even when a particular buffer in the two port RAM is mapped into host memory. However, when the buffer is mapped into the security device's address space, the host computer has no access to the buffer.

General Operation

The overall operation of the system will now be discussed with reference to Fig. 13. The first step is to configure the network. A security officer 502 at the network security center 500 interacts via menus at the security center console 504 to define security devices (510, 520), principals (512, 522), classification levels, etc. This includes profiles for principals, which includes associations such as the ability of host 514 to transmit to host 524.

The security officer 502 also specifies the authentication method (card, password, etc.) to be used by the principal. If the principals are to use an authentication card, the security officer creates individual authentication cards at the card reader 506 and gives these cards to principals (512, 522). During this process, the security officer 502 creates an

IPR2024-00027
CrowdStrike Exhibit 1002 Page 563

initialization card for security devices (510, 520) and physically goes to that site to read in the card via card readers (516, 526) in order to complete installation of the devices.

Each principal goes to the location of the security device (510, 520) and initiates the authentication method specified for the principal. Different principals may be configured to use different means, for example, principal 512 may be configured to use an authentication card and principal 522 may be configured to use a password.

When an authentication card is required, the principal 512 inserts the authentication card into the card reader 516, which is connected to the security device 510. The card is read by the security device 510. Principal 522, which has a password instead of a card, types the password at the console of host 528. The host transfers the password to the security device 524 via a trusted path.

The device will be defined to support a specified authentication type. The principal who is attempting to use the device must have the appropriate authentication data. The security devices independently transfer the authentication data to the security center 500 via an encrypted (nominally Triple DES, although other methods are conceivable) connection. The security center replies to the security device, listing the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 564

profiles that may be selected by the principal. This may be anywhere from a single profile up to a hundred profiles.

Each principal (independently) selects the desired profile at the keypad of the local card reader (516, 526). Each security device (510, 520) sends this information to the security center 500 via an encryption connection.

There is not an initial communication without the selected profile between the device and the controller. The device gathers the required data and then sends all of the information via the encrypted channel to the controller. The security center responds to the security device with the security windows to be used for MAC and the security associations to be used for DAC. Assuming that security devices (510, 520) permit associations between hosts (518, 528), the security devices negotiate traffic keys between the two devices (based on a shared secret value downloaded from the security center). The shared secret is only one of the possible mechanisms. The approved mechanisms are defined in the IPSec standards.

At this point the two hosts (518, 528) may communicate via their respective security devices, provided that communications are consistent with the predefined security windows (MAC).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 565

Example Embodiment

Turning to Fig. 14, an example of a local area network 10 is shown connected to a network 30. The LAN 10 comprises a first VPN 80, and a second VPN 82. The VPNs 80, 82, are established by a security officer that sets up permitted associations between hosts based on IP addresses. The VPNs are collections of host IP addresses that are permitted to communicate. Hosts on the second VPN 82 are invisible to hosts which are not directly addressable. This centralized configuration of secure VPNs is enforced by the respective security devices 18 of each host.

In the example set forth in Fig. 14, working from the left of the page, a first host 86 is configured to operate on both the first and second VPNs 80, 82. The third and fourth hosts, 90, 92, only have the ability to operate on either the first VPN 80 or the second VPN 82, respectively, but cannot access both VPNs. This is merely a matter of defining the IP associations permitted for this host.

The second host 88 has a security device 18 that permits the host 88 to operate either on the second VPN 82 or on an untrusted line 84. This ability to switch between a trusted network 82 and an untrusted network 84 is defined by the security officer at the NSC by defining multiple permitted profiles for

IPR2024-00027
CrowdStrike Exhibit 1002 Page 566

a principal. When the principal authenticates at the security device 18 associated with this host 88, the principal determines which of the permitted profiles is to be used.

According to our example, the profile for the principal operating at the second host 88 is twofold. First, for the second VPN 82, the host 88 is permitted to transmit and receive associations with the first host 86 and with the fourth host 92. A security window for MAC is also defined. A security window is always used, except when the security device is operating in a bypass mode to unprotected hosts, such as hosts without a security device. Secondly, when operating on network 84, the second host 88 is permitted to communicate with any unprotected host (not shown) anywhere on the Internet.

Although the principal at host 88 may switch between the second trusted VPN 82, and an untrusted path 84, the principal may only use one profile at a time. Thus, the host 88 may connect to either the untrusted network 84 or the trusted network 82 to communicate only with hosts 86, 92. Thus, it is impossible for a host on the Internet to route packets through hosts 88 and back out on the trusted network 82, perhaps to attack the fourth host 92.

If the principal has a profile that permits communication with unprotected hosts (such as hosts that do not

IPR2024-00027
CrowdStrike Exhibit 1002 Page 567

have a security device 18), the security device operates in a "bypass" mode. In the "bypass" mode, the security device does not provide encryption but does implement MAC and DAC. Thus, MAC and DAC are always performed, though encryption may not be performed depending on the destination node.

Further to this example, if the second host 88 is an untrusted host and it is operating at a secret level, then the memory of the second host 88 before permitting connections to or from any other untrusted host. This is possible, for instance, by switching disk drives, such as a slide-in drive, and power-cycling the host to clear memory.

Unlike the second host 88, the third host 90 is configured (by the user's profile) to be able to connect over the Internet 30 to a remote host 94, as well as to the first host 86 via line 80. Simultaneously, the first host 86 may be connected to the fourth host 92 via line 82.

Alternative Embodiments

The preferred embodiment is for networks including multi-user servers, where the principal is not necessarily a user of the attached host. However, the invention may be used in any suitable network environment, such as one having single-user workstations where the principal is the current user of the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 568

workstation. The security devices in the single-user workstation network may be configured so that the classification of the host is related to the administrative clearance of the user/principal operating at the host. The network would support multi-level security for communications between individual users.

Although the preferred embodiment is also for a network having classified information, the invention has uses for unclassified environments as well. As implemented in an unclassified network, for instance, instead of establishing different hierarchical security levels, various non-hierarchical descriptors may be defined by the network security officer. Also, a combination of levels and descriptors may be defined.

In the present embodiment, for example, the security device may be used to monitor and distribute incoming and outgoing information in accordance with the various descriptors. Principals are assigned profiles that define permitted associations and the like. Each principal may be assigned one or more profiles. The principal may log in at any host and select from the one or more profiles. The selected user profile is then used to define the permitted communications for that host. Accordingly, a principal may use any host to connect to the network, and select a user profile to establish the parameters for that connection.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 569

In yet another alternative embodiment, the security device 18 may be eliminated altogether and the security mechanisms implemented by software located at the computer (or as otherwise suitable). Thus, for instance, the software is configured to implement encryption, DAC and MAC for all incoming and outgoing communications. In addition, the software establishes and implements user profiles, association lists, and audit events, as defined by the network security officer.

The foregoing descriptions and drawings should be considered as illustrative only of the principles of the invention. The invention may be configured in a variety of manners and is not limited by the design of the preferred embodiment. Numerous applications of the present invention will readily occur to those skilled in the art. Therefore, it is not desired to limit the invention to the specific examples disclosed or the exact construction and operation shown and described. Rather, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 570

We Claim:

1. A security device for connecting a host computer from a host bus to a computer network, the security device comprising a local bus, a network interface connecting said local bus to the computer network, and a two-port memory device connecting said local bus to the host bus.

2. The security device of claim 1, wherein the two-port memory device has two bus interfaces, a first interface for communicating with the host bus and a second interface for communicating with the local bus.

3. The security device of claim 2, wherein information to be passed between the host bus and the local bus can not be simultaneously located on the first interface and the second interface.

4. The security device of claim 2, wherein information to be transmitted from a sending host to a receiving host is written from the host bus to the first interface, then read from the first interface to the second interface.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 571

5. The security device of claim 1, wherein the two-port memory device comprises a two-port RAM.

6. The security device of claim 1, further comprising an internal system memory connected to said local bus for storing information for said firmware and said interface.

7. The security device of claim 1, further comprising a cipher unit connected to the local bus.

8. The security device of claim 1, further comprising an authentication interface unit for authenticating a computer user.

9. The security device of claim 1, wherein said interface comprises a network coprocessor.

10. The security device of claim 1, wherein the network comprises a local area, Ethernet or token ring network.

11. The security device of claim 1, further comprising a central processing unit for implementing firmware.

12. The security device of claim 1, wherein security is implemented at a network layer of protocol hierarchy.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 572

13. A method for controlling a sending computer to transmit information to a receiving computer over a computer network, the method comprising:

receiving the information to be transmitted to the receiving computer from the sending computer;

implementing security mechanisms to determine whether communication is authorized from the sending computer to the receiving computer and, if not, then terminating the transmission of information and, if so, then encrypting the information to be transmitted; and,

transmitting the encrypted information to the receiving computer over the computer network.

14. The method of claim 13, wherein the step of implementing security mechanisms comprises the steps of determining if the receiving computer is in a transmit list and consistent with a transmit security window and, if both conditions are not satisfied then terminating the transmission of information, otherwise encrypting the information to be transmitted.

15. The method of claim 14, wherein the steps of determining if the receiving computer is in a transmit list and consistent with a transmit security window comprises the steps of performing

IPR2024-00027
CrowdStrike Exhibit 1002 Page 573

discretionary access control and mandatory access control, respectively.

16. The method of claim 13, further comprising the step of generating an audit in addition to terminating the flow of information.

17. The method of claim 13, wherein security is implemented at a network layer of protocol hierarchy.

18. The method of claim 19, the method being implemented by security devices, one security device connected to each one the sending computer and the receiving computer.

19. A method for controlling a receiving computer to receive information transmitted from a transmitting computer over a computer network, the method comprising:

receiving the information to be received by the receiving computer from the computer network;

implementing security mechanisms to determine whether communication is authorized from the sending computer to the receiving computer and, if not, then terminating the transmission

IPR2024-00027
CrowdStrike Exhibit 1002 Page 574

of information and, if so, then decrypting the information to be received; and,

transmitting the decrypted information to the receiving computer for reception thereof.

20. The method of claim 19, wherein the step of implementing security mechanisms comprises the steps of determining if the transmitting computer is in a receive list and consistent with a receive security window and, if both conditions are not satisfied then terminating the transmission of information, otherwise decrypting the information to be received.

21. The method of claim 20, wherein the steps of determining if the transmitting computer is in a receive list and consistent with a receive security window comprises the steps of performing discretionary access control and mandatory access control, respectively.

22. The method of claim 19, further comprising the step of generating an audit in addition to terminating the flow of information.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 575

23. The method claim 19, wherein security is implemented at a network layer of protocol hierarchy.

24. The method of claim 19, the method being implemented by security devices, one security device connected to each one the sending computer and the receiving computer.

25. A secure network having a plurality of host computers accessible to users and connected to a network medium that has access to an untrusted line, the secure network comprising:

a network security controller for enabling a security officer to generate at least one user profile for each user, each user profile defining at least one destination which the user is authorized to access; and,

security devices connected to the network medium for receiving the user profiles generated at the network security controller, each security device associated with one host computer, each security device having an authorization device for authorizing users at the associated host computer, the security device permitting the authorized user, via the associated host computer, to select a user's profile associated with the user and for restricting access of the host computer to the at least one destination defined in the selected user's profile.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 576

26. The network of claim 25, wherein the at least one destination comprises at least one other host computer of the network or the untrusted line.

27. The network of claim 25, the security device implementing security mechanisms when the host computer connects to a trusted destination.

28. The network of claim 25, the security device not implementing security mechanisms when the host computer connects to an untrusted destination.

29. The network of claim 25, wherein the untrusted line comprises the Internet.

30. The network of claim 25, wherein a user cannot simultaneously communicate with a trusted destination and an untrusted destination.

31. The network of claim 25, wherein a user is prevented from simultaneously connecting to destinations having different security levels.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 577

32. The network of claim 25, wherein a user can only select one profile at a time.

33. The network of claim 25, wherein the user profiles define virtual private networks of communication comprising subsets of host computers.

34. The network of claim 25, wherein security is implemented at a network layer of protocol hierarchy.

35. The network of claim 25, wherein at least one user profile has only one destination.

36. The network of claim 25, wherein the destination in a user's profile correspond to a level of security granted the user.

37. The network of claim 25, wherein the security devices are integrated with the associated host computer.

38. A method for operating a network having a plurality of host computers accessible to users and connected to a network medium that has access to an untrusted line, the method comprising:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 578

generating at least one user profile for each user, each user profile defining at least one destination which the user is authorized to access;

authorizing a user at a host computer;

determining, at the host computer, the at least one user profile associated with the authorized user;

permitting, at the host computer, the authorized user to select a user's profile associated with the user; and

restricting access of the host computer to the at least one destination defined in the selected user's profile.

39. The method of claim 38, wherein the at least one destination comprises at least one other host computer of the network or the untrusted line.

40. The method of claim 38, further comprising the step of implementing a security mechanism when the host computer connects to a trusted destination.

41. The method of claim 38, further comprising the step of not implementing security mechanisms when the host computer connects to an untrusted destination.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 579

42. The method of claim 38, wherein the untrusted line comprises the Internet.

43. The method of claim 38, wherein a user cannot simultaneously communicate with a trusted destination and an untrusted destination.

44. The method of claim 38, wherein a user is prevented from simultaneously connecting to destinations having different security levels.

45. The method of claim 38, wherein a user can only select one profile at a time.

46. The method of claim 38, wherein the user profiles define virtual private networks of communication comprising subsets of host computers.

47. The method of claim 38, wherein security is implemented at a network layer of protocol hierarchy.

48. The method of claim 38, wherein at least one user profile has only one destination.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 580

49. The method of claim 38, wherein the destination in a user's profile correspond to a level of security granted the user.

50. A multi-level secure network having a plurality of host computers accessible to users and connected to a network medium that has access to an untrusted line, the secure network comprising a security device coupled between at least one host computer and the network medium which operates at a network layer communications protocol and a network security controller for controlling the security device to establish connections to the network medium.

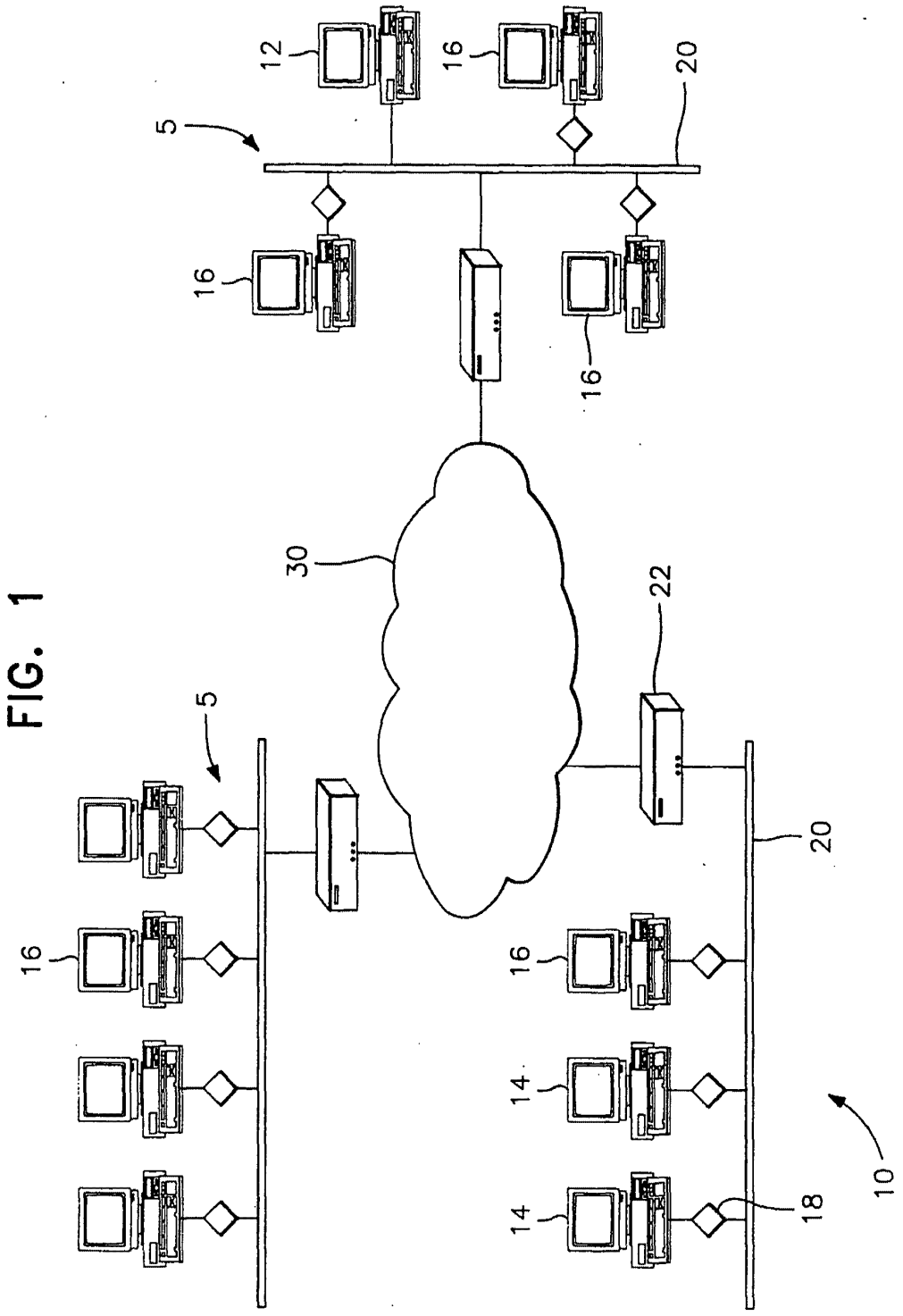
51. The multi-level secure network of claim 50, wherein the network security controller audits events.

52. The multi-level secure network of claim 50, wherein the security device prevents simultaneous connection to a trusted line and an untrusted line.

53. The multi-level secure network of claim 50, wherein the security device prevents simultaneous connection between lines of different security levels.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 581

FIG. 1



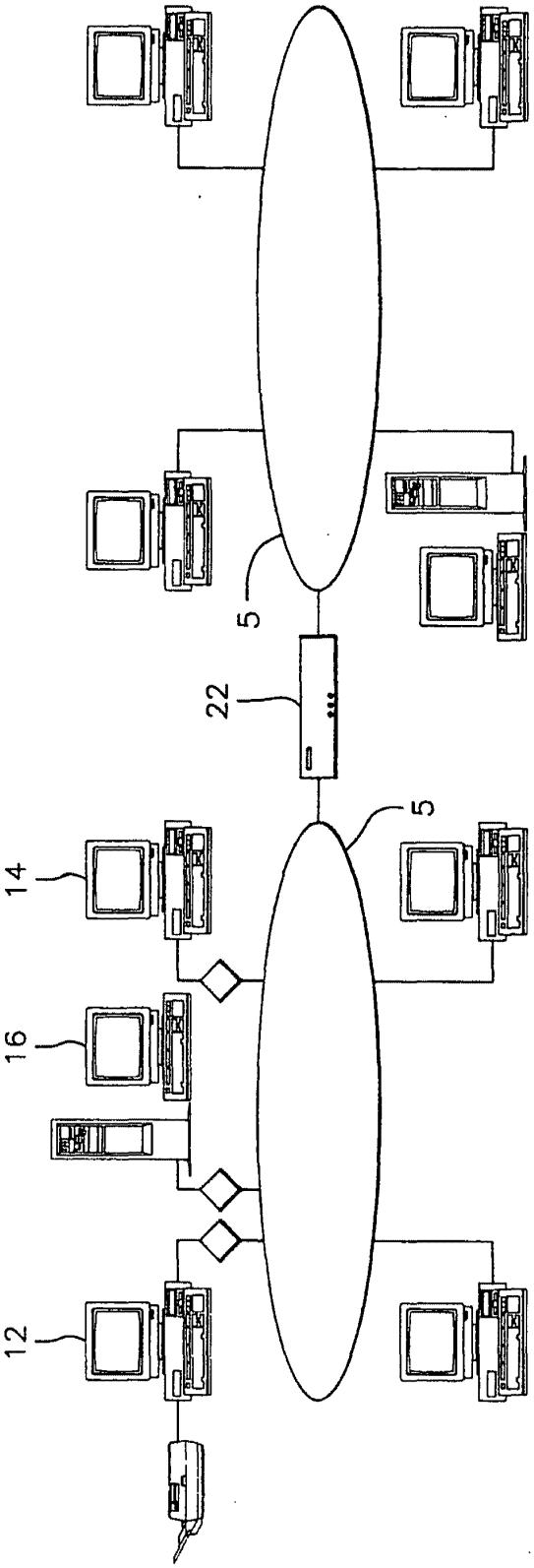
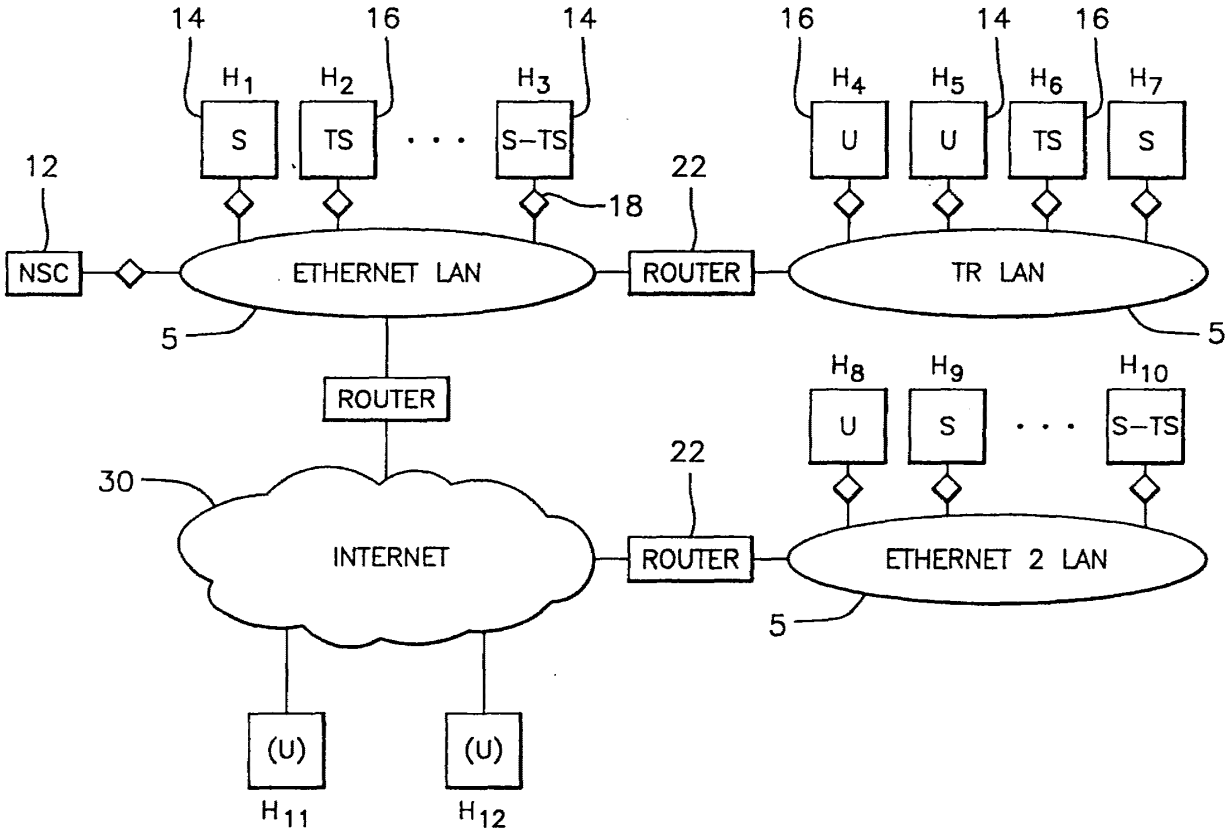


FIG. 2

FIG. 3



IPR2024-00027
CrowdStrike Exhibit 1002 Page 584

FIG. 4

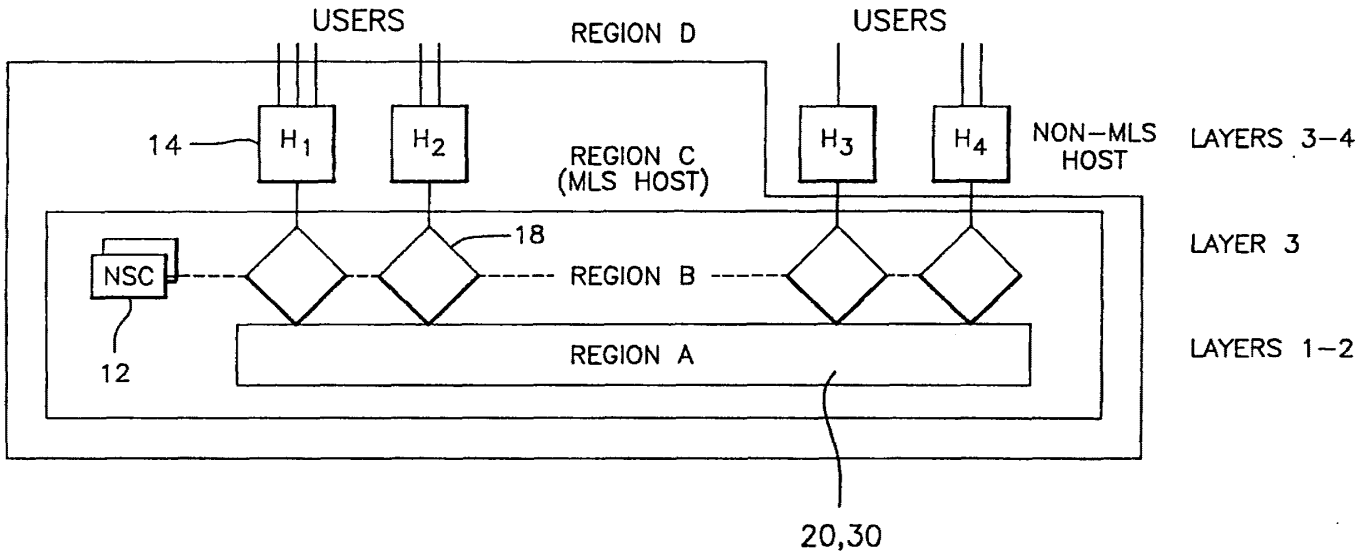
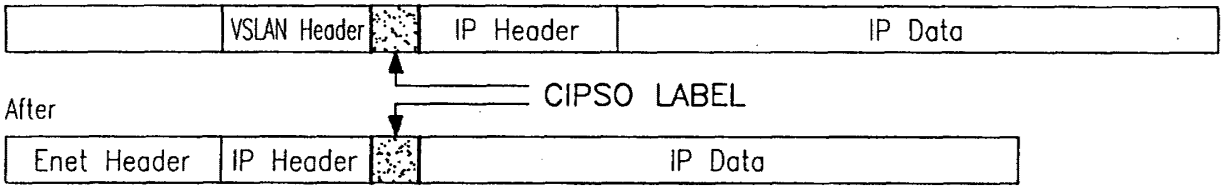

USERS
REGION D
USERS
14
H1
H2
H3
H4
NON–MLS HOST
LAYERS 3–4
REGION C (MLS HOST)
18
LAYER 3
NSC
REGION B
REGION A
LAYERS 1–2
12
20,30

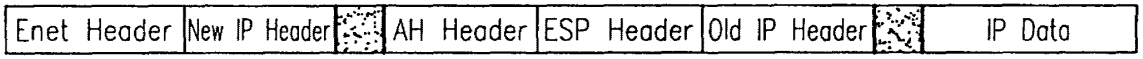
FIG. 5

* CIPSO LABELING



* HOST TO HOST COMMUNICATION

Travel log



Transport



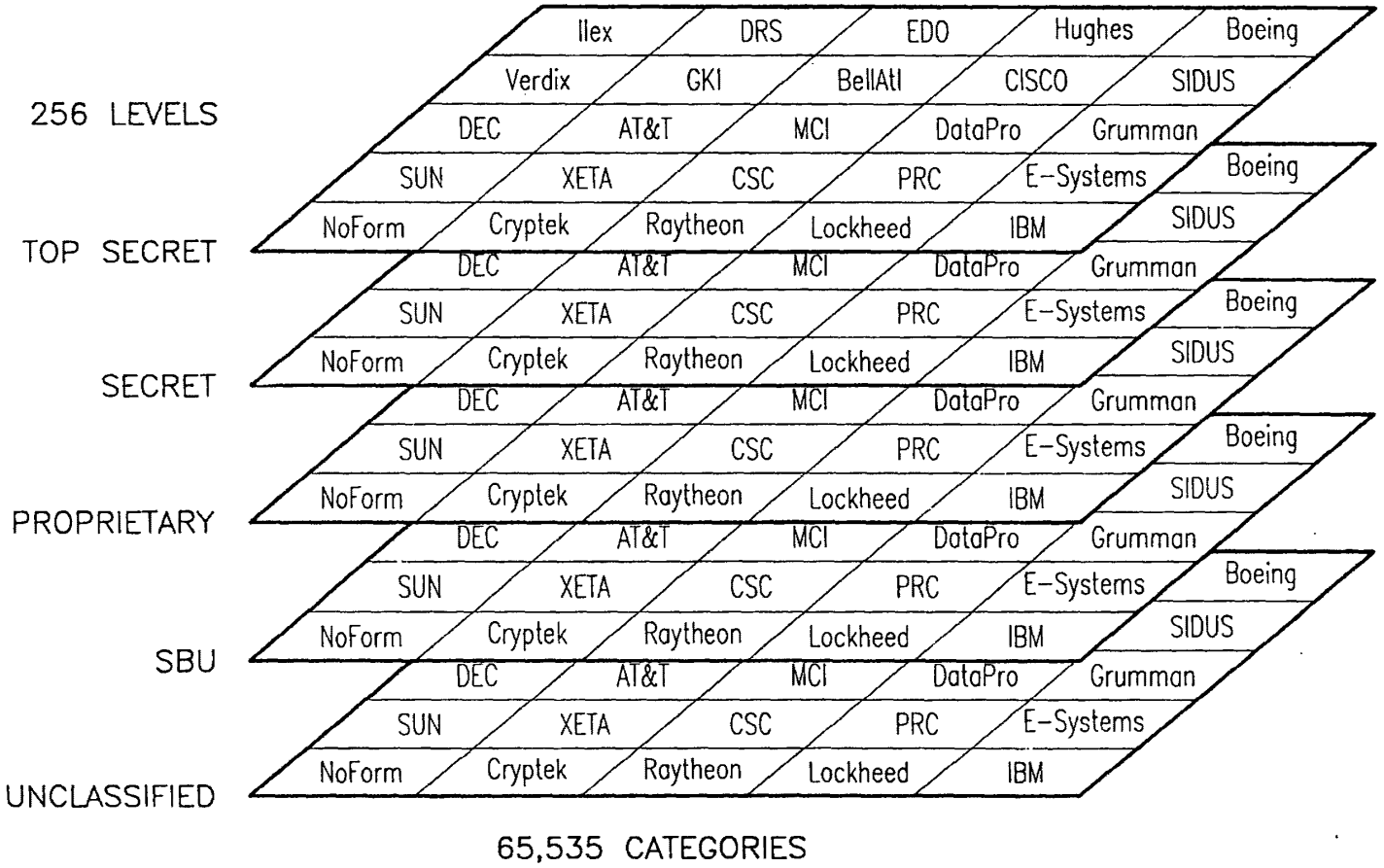
* CONTROL COMMUNICATION



After



FIG. 6



IPR2024-00027
CrowdStrike Exhibit 1002 Page 587

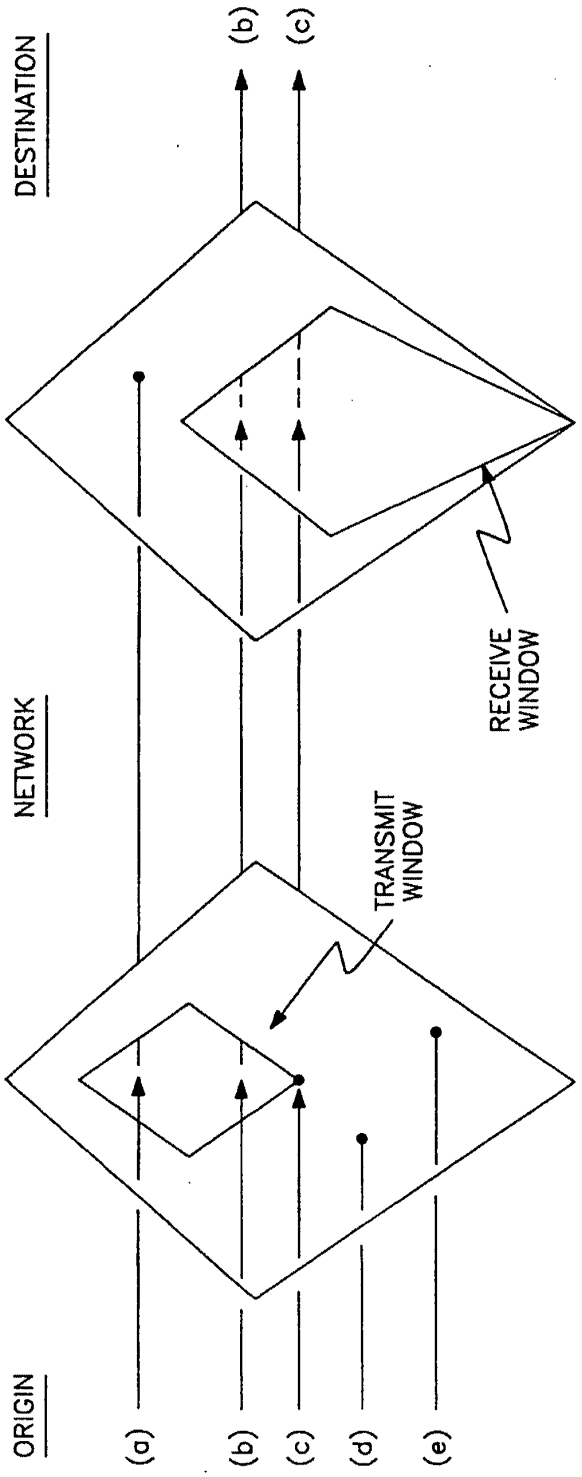

DESTINATION
(b)
(c)
RECEIVE WINDOW
NETWORK
TRANSMIT WINDOW
ORIGIN
(a)
(b)
(c)
(d)
(e)

FIG. 7

FIG. 8

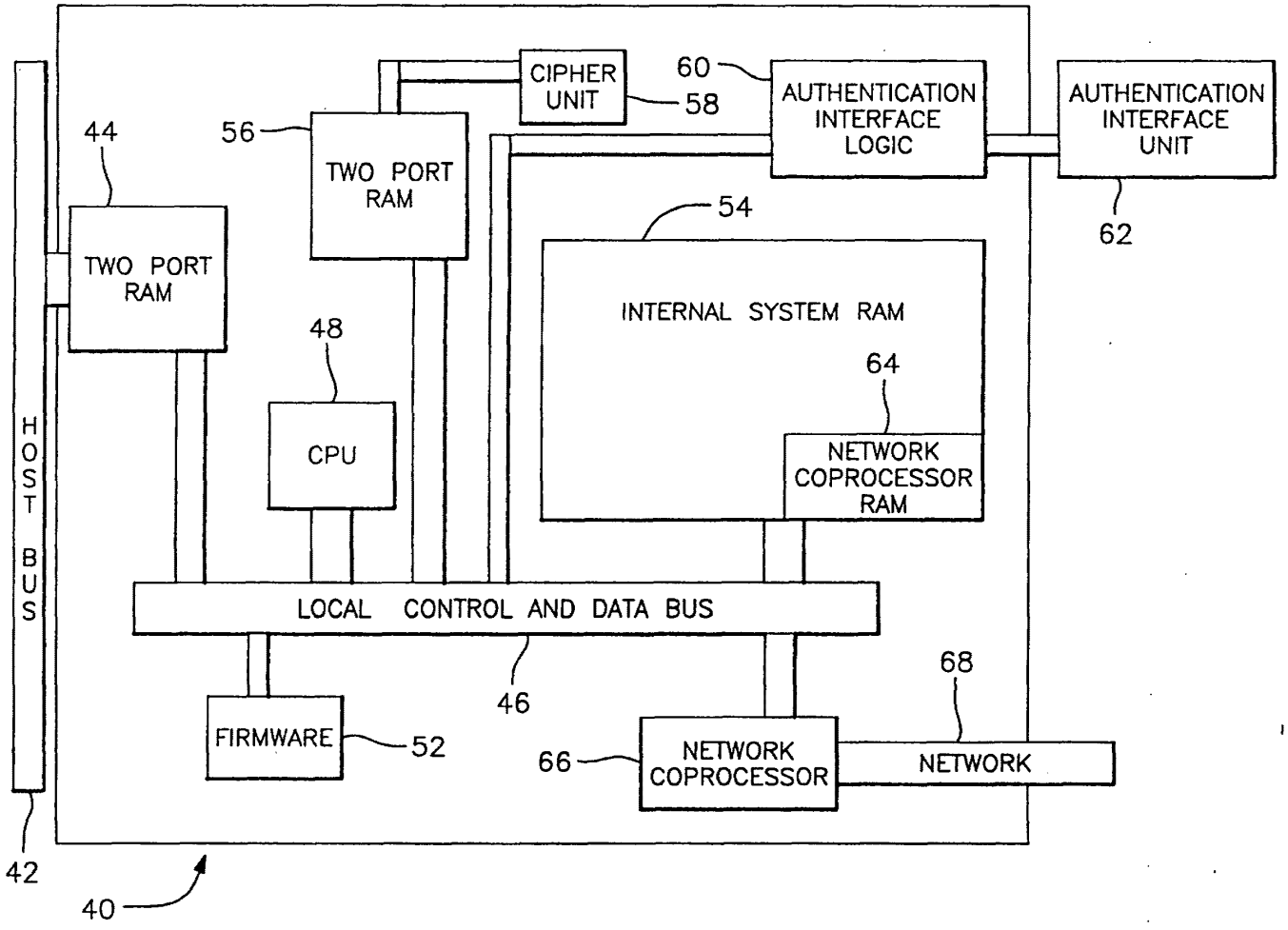

CIPHER UNIT
60
58
AUTHENTICATION INTERFACE LOGIC
AUTHENTICATION INTERFACE UNIT
62
44
56
TWO PORT RAM
TWO PORT RAM
54
INTERNAL SYSTEM RAM
64
48
CPU
NETWORK COPROCESSOR RAM
HOST BUS
LOCAL CONTROL AND DATA BUS
46
68
FIRMWARE
52
66
NETWORK COPROCESSOR
NETWORK
42
40

FIG. 9

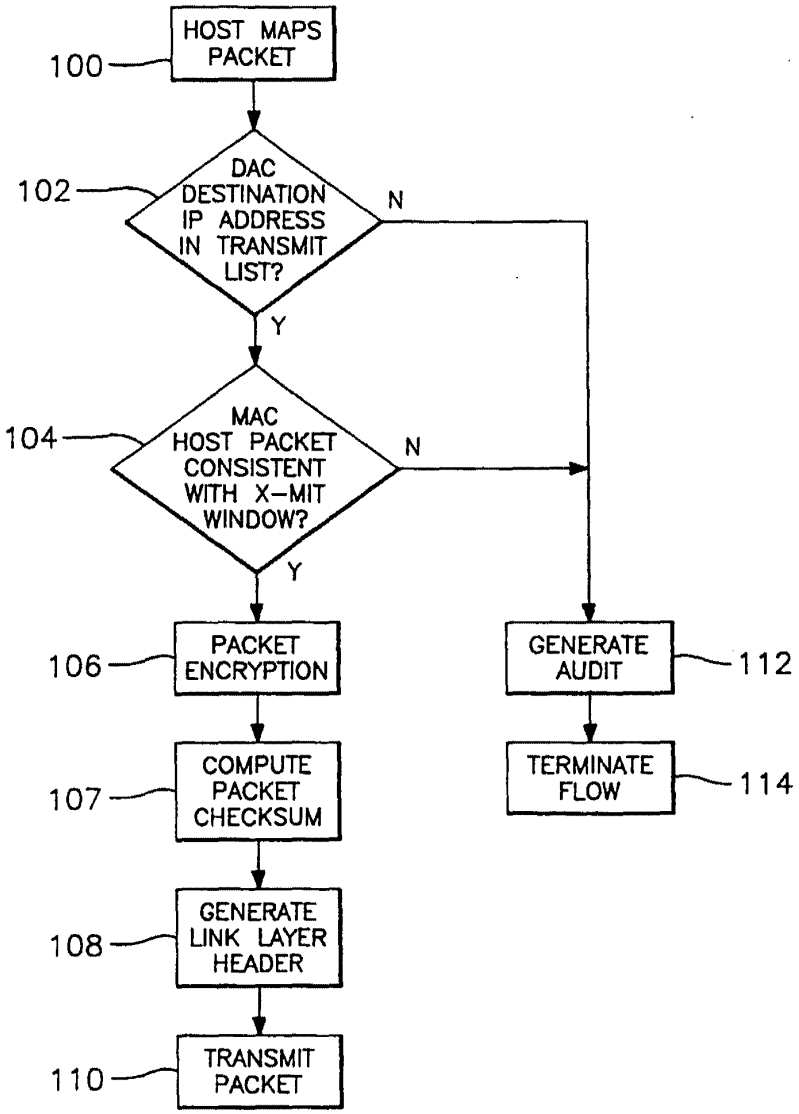

HOST MAPS PACKET
100
DAC DESTINATION IP ADDRESS IN TRANSMIT LIST?
102
N
Y
MAC HOST PACKET CONSISTENT WITH X-MIT WINDOW?
104
N
Y
PACKET ENCRYPTION
106
COMPUTE PACKET CHECKSUM
107
GENERATE LINK LAYER HEADER
108
TRANSMIT PACKET
110
GENERATE AUDIT
112
TERMINATE FLOW
114

FIG. 10

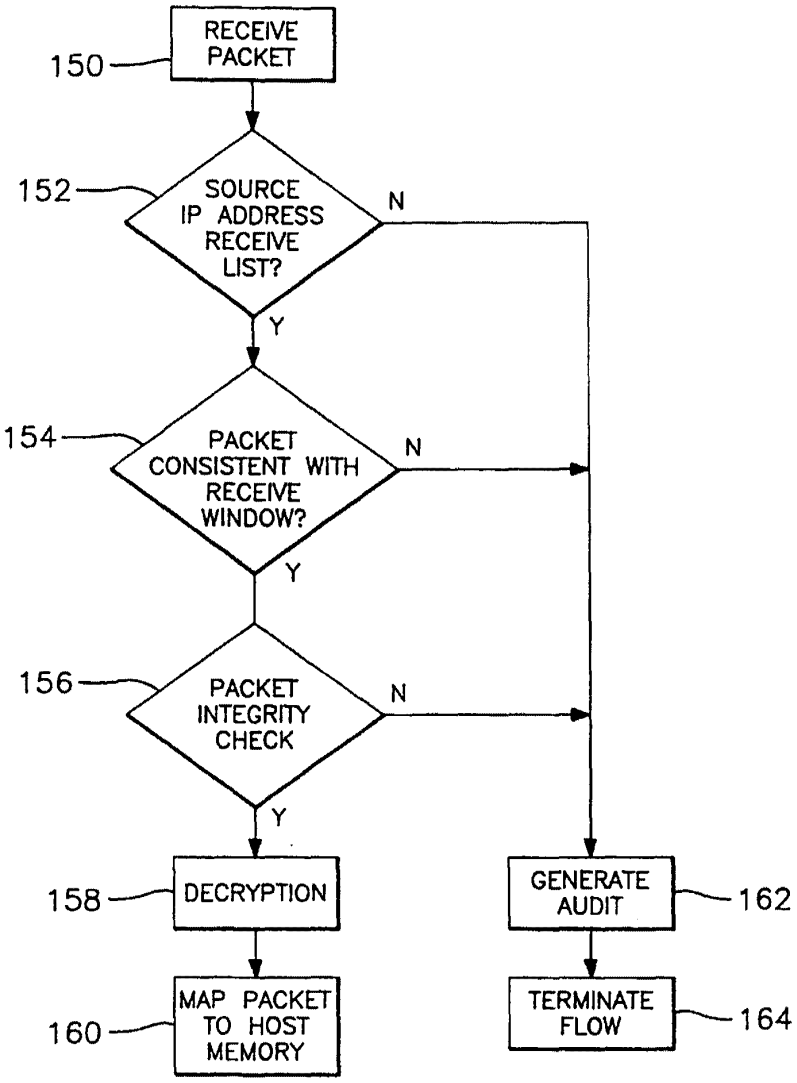

RECEIVE PACKET
150
SOURCE IP ADDRESS RECEIVE LIST?
152
N
Y
PACKET CONSISTENT WITH RECEIVE WINDOW?
154
N
Y
PACKET INTEGRITY CHECK
156
N
Y
DECRYPTION
158
MAP PACKET TO HOST MEMORY
160
GENERATE AUDIT
162
TERMINATE FLOW
164

FIG. 11
(PRIOR ART)

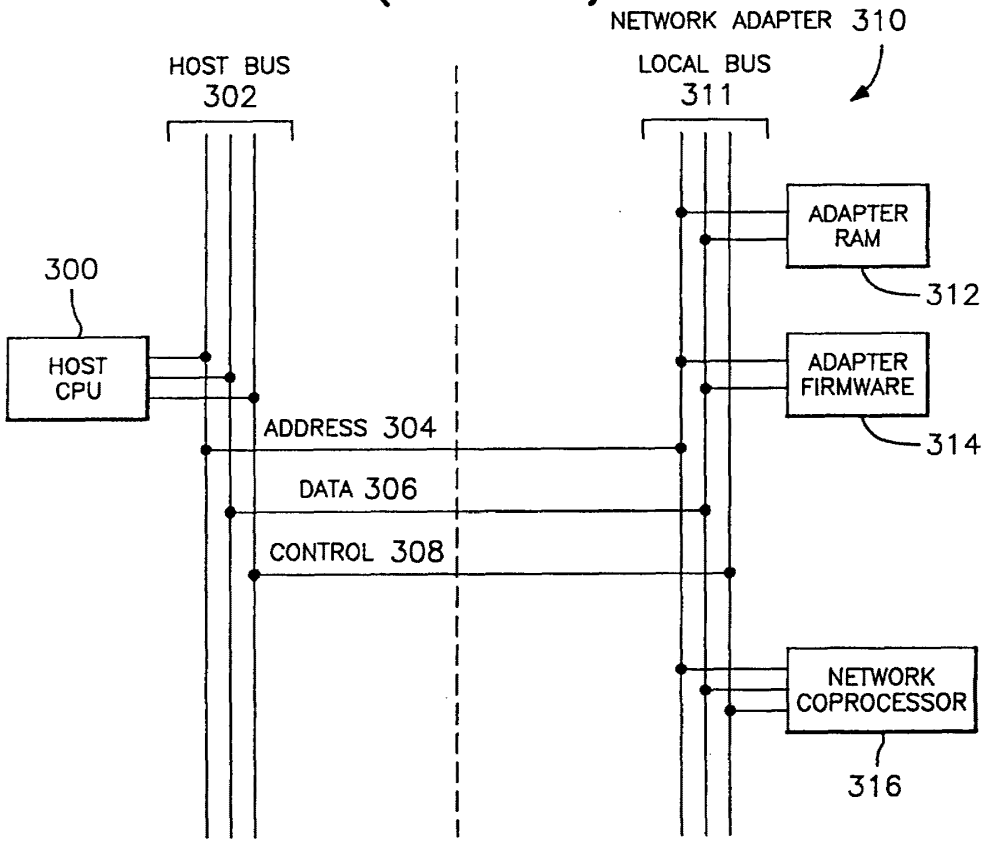


FIG. 12

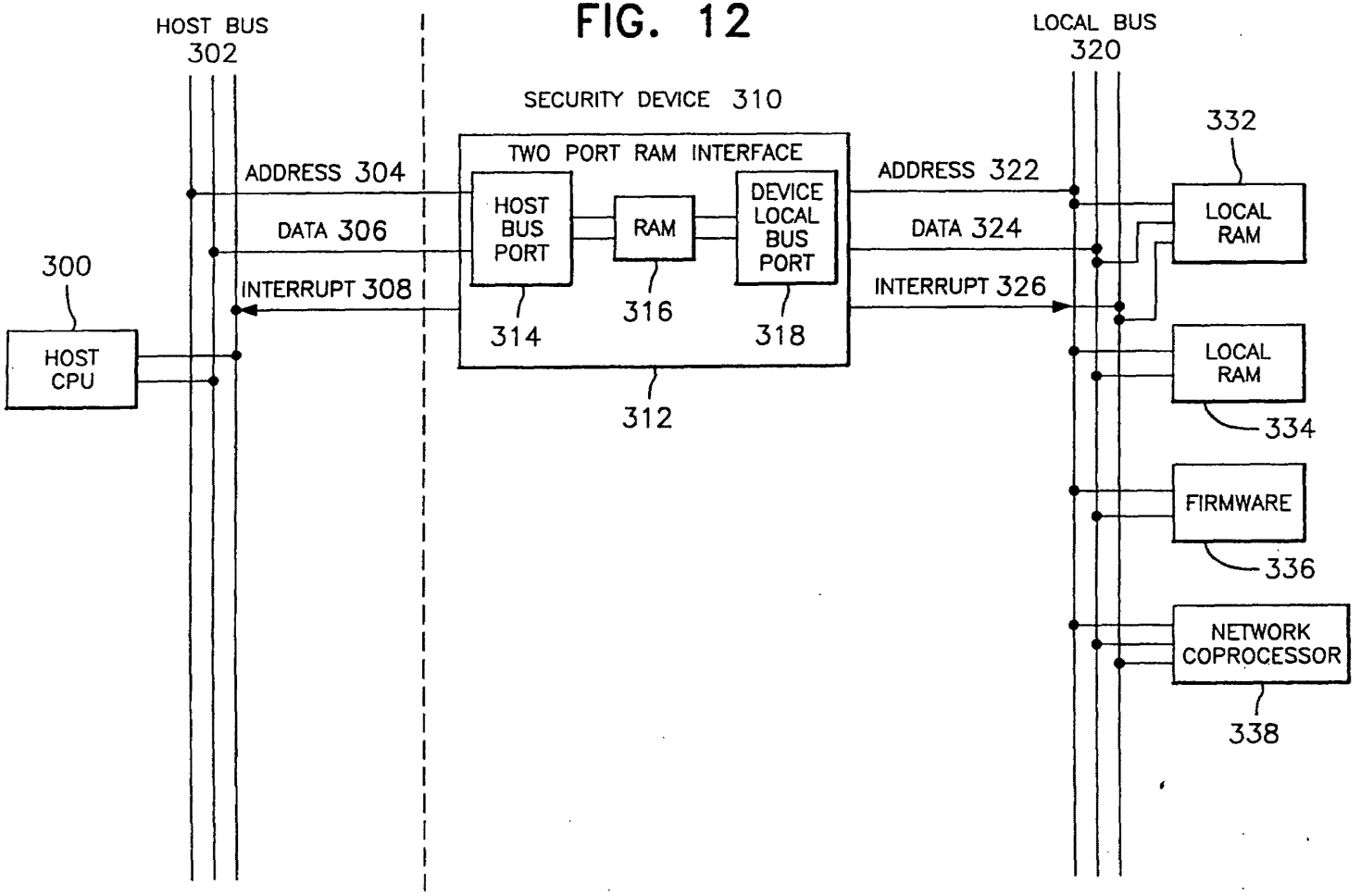

HOST BUS
302
SECURITY DEVICE 310
TWO PORT RAM INTERFACE
ADDRESS 304
DATA 306
INTERRUPT 308
HOST BUS PORT
RAM
DEVICE LOCAL BUS PORT
314
316
318
312
300
HOST CPU
LOCAL BUS
320
ADDRESS 322
DATA 324
INTERRUPT 326
332
LOCAL RAM
LOCAL RAM
334
FIRMWARE
336
NETWORK COPROCESSOR
338

FIG. 13

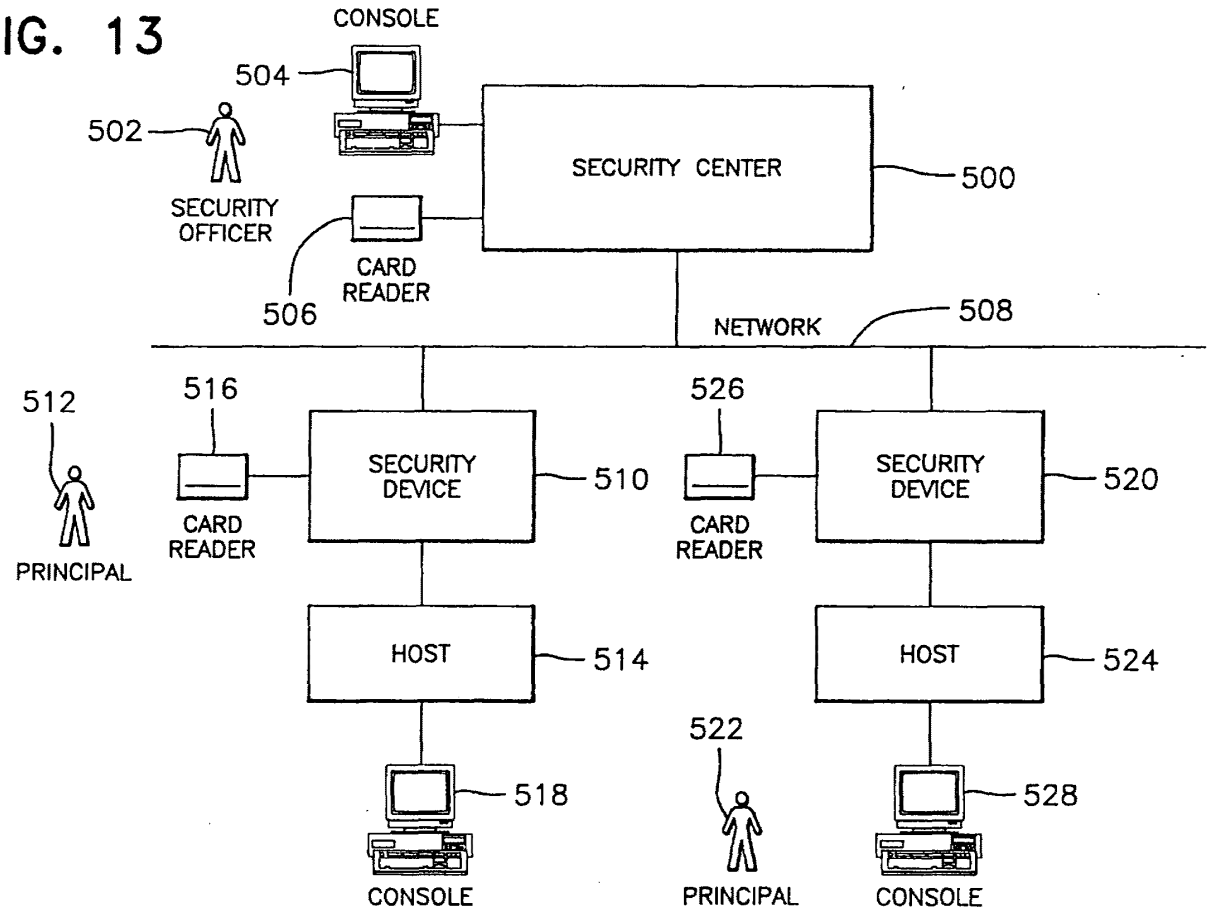
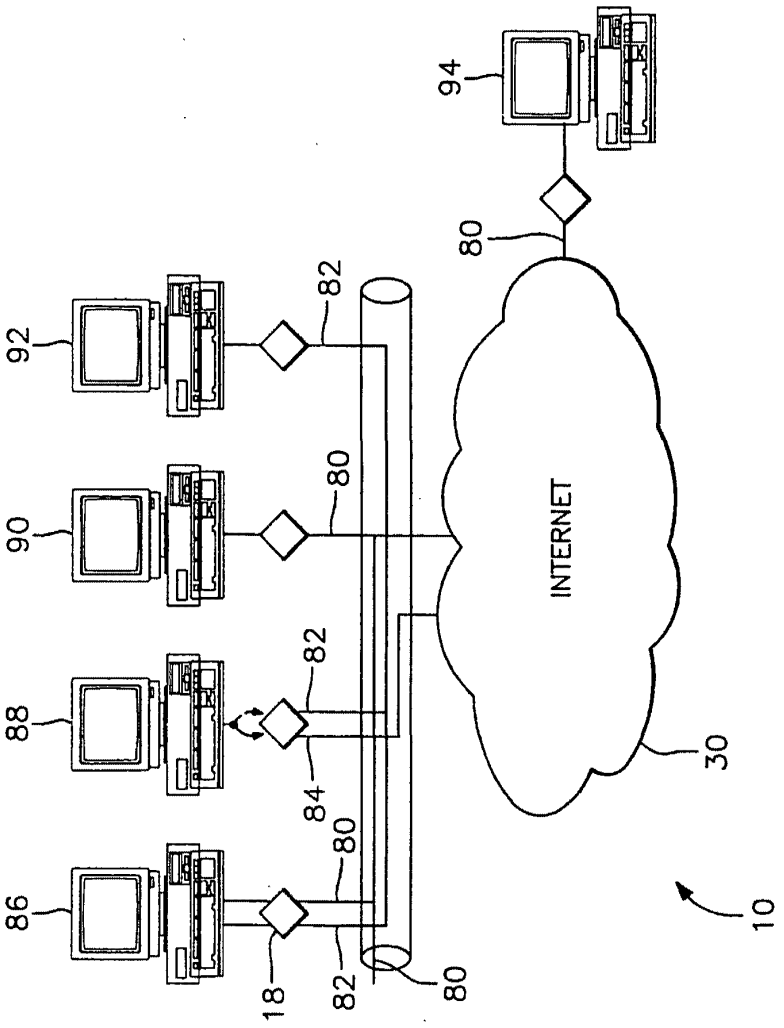

CONSOLE
504
502
SECURITY OFFICER
SECURITY CENTER
500
CARD READER
506
NETWORK
508
512
516
526
PRINCIPAL
CARD READER
SECURITY DEVICE
510
CARD READER
SECURITY DEVICE
520
HOST
514
HOST
524
522
518
528
PRINCIPAL
CONSOLE
CONSOLE

FIG. 14



PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6: G06F 12/14 — A1

(11) International Publication Number: **WO 00/25214**

(43) International Publication Date: 4 May 2000 (04.05.00)

(21) International Application Number: PCT/US99/25455

(22) International Filing Date: 28 October 1999 (28.10.99)

(30) Priority Data:

60/105,963	28 October 1998 (28.10.98)	US
09/248,788	12 February 1999 (12.02.99)	US

(71) Applicant: SECURESOFT, INC. [US/US]; Suite 520, 275 Shoreline Drive, Redwood Shores, CA 94065 (US).

(72) Inventors: MORICONI, Mark; 11 Michaels Way, Atherton, CA 94027 (US). QIAN, Shelly; 20643 Greenleaf Drive, Cupertino, CA 95014 (US).

(74) Agents: TOCZYCKI, Robert et al.; Carr & Ferrell LLP, Suite 200, 2225 East Bayshore Road, Palo Alto, CA 94303 (US).

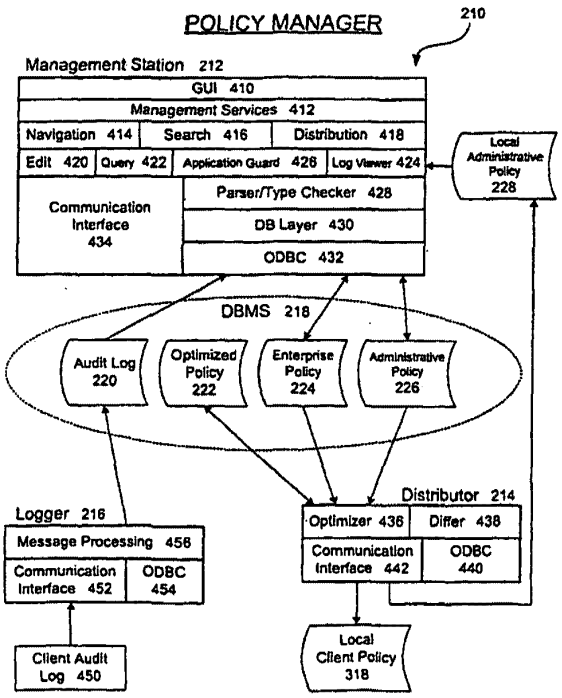
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: MAINTAINING SECURITY IN A DISTRIBUTED COMPUTER NETWORK

(57) Abstract

A system and method for maintaining security in a distributed computing environment comprises a policy manager (210) located on a server for managing and distributing a security policy, and an application guard (426) located on a client for managing access to securable components as specified by the security policy. In the preferred embodiment, a global policy specifies access privileges of the user to securable components. The policy manager (210) may then preferably distribute a local client policy (318) based on the global policy to the client. An application guard (426) located on the client then manages access to the securable components as specified by the local policy (318).



IPR2024-00027
CrowdStrike Exhibit 1002 Page 596

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

MAINTAINING SECURITY IN A DISTRIBUTED COMPUTER NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to, and claim priority in, co-pending U.S. Provisional Patent Application Serial No. 60/105,963, entitled "System And Method For Maintaining Security In A Distributed Computer Network," filed on October 28, 1998.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer security systems, and relates more particularly to a system and method for managing and enforcing complex security requirements in a distributed computer network.

2. Description of the Background Art

Computer security issues have become more complex with the continual evolution of contemporary computer systems. As corporations utilize increasingly distributed and open computing environments, the security requirements of an enterprise typically grow accordingly. The complexity of employee, customer and partner access to critical information assets, while assuring proper security, has proven to be a major hurdle. For example, many organizations deploy applications that allow their external business partners, as well as their own internal employees, to access sensitive information resources within the enterprise. In the absence of adequate security measures, an enterprise may thus be subject to the risk of decreased security and confidentiality.

While most organizations focus their security concerns on protecting the internal network from the outside world, it is estimated that 80-90% of all corporate security breaches come from within an organization (source: Aberdeen Group, September 1997).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 598

This further underscores the need to specify and enforce an access control security policy within the enterprise network.

In today's complex business environment, specifying, stating, implementing and managing an enterprise access control policy may be both difficult and inefficient. When corporate data and applications revolved around a mainframe model, the problem of defining and managing access to corporate applications was relatively straightforward. Today, the complexity of business methods, as well as the complexity of distributed application architectures, may force companies to resort to manual, ineffective or highly custom approaches to access control in their attempts to implement the business process.

To secure a complex and distributed computer system, the system may typically employ a combination of encryption, authentication, and authorization technologies. Encryption is a means of sending information between participants in a manner that prevents other parties from reading the information. Authentication is a process of verifying a party's identity. Authorization is a technique for determining what actions a participant is allowed to perform.

Encryption and authentication are well-understood and have led to effective network security products, whereas authorization technology is not as well developed, and is often inadequate for many enterprises. The security approach of most companies today is to focus on the authentication of users to ensure that those users are part of the organization or a member of a select group. Authentication can be accomplished with a number of different approaches, from simple password or challenge response mechanisms to smart cards and biometric devices such as a fingerprint reader. Once users are authenticated, however, there is still a significant problem in managing and enforcing their set of privileges, which may be unique and vary widely between users. The same authentication mechanism can be used for every user, but different authorization mechanisms must be developed for most applications. Therefore, reliable and efficient access control is a much more difficult problem facing enterprises today.

Authentication mechanisms often work together with some sort of access control facility that can protect information resources from unauthorized users. Examples of network security products include firewalls, digital certificates, virtual private networks, and single sign-on systems. Some of these products provide limited support for resource-level authorization. For example, a firewall can screen access requests to an

IPR2024-00027
CrowdStrike Exhibit 1002 Page 599

application or a database, but does not provide object-level authorization within an application or database. Single Sign-On (SSO) products, for example, maintain a list of resources an authenticated user can access by managing the login process to many different applications. However, firewalls, SSO and other related products are very limited in their ability to implement a sophisticated security policy characteristic of many of today's enterprises. They are limited to attempting to manage access at a login, or "launch level", which is an all or nothing approach that inherently cannot implement a business-level policy.

A real-world security policy within a large enterprise is a detailed and dynamic knowledge base specific to that organization. The authorization privileges are specific to the constantly evolving set of users, applications, partners, and global policies that the enterprise puts in place to protect its key information resources. A security policy within a large enterprise can consist of tens or hundreds of thousands of individual rules that cover which users are authorized to access particular applications, perform various operations, or manage the delegation and transfer of tasks. Many of these policy rules that implement the business practice of the organization have to be hard coded within custom-built applications or stored in the database.

The key problem is that these policy rules are localized, scattered throughout the organization, and embedded in applications and databases. Such embedding is expensive and error-prone, and mitigates against efficient policy updates. An organization cannot effectively implement and manage the resulting policy. Inconsistencies arise and updates can quickly become unmanageable. Policy queries and analysis from a global perspective are nearly impossible. The resulting policy begins to diverge from the intended business practices of the organization. Compromises are made in the policy implementation at the department level, and auditors can quickly become frustrated.

The increasing security risks associated with the proliferation of distributed computing, including Intranet and Extranet applications, are prompting many organizations to explore a broad range of security solutions for controlling access to their important information assets. Although organizations have a number of solutions to choose from for authenticating users (determining and verifying who is attempting to gain access to the network or individual applications), there is little choice when it comes to controlling what users can do and when they can do it to the extent necessary to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 600

implement the kinds of complex security policies required by modern organizations. Organizations have been forced to choose between custom authorization solutions that are costly, error-prone, and difficult to manage, or third-party solutions that are very limited in their ability to control access to information across applications and databases.

A real-world security policy within a large organization is a detailed and dynamic knowledge base that determines which users are authorized to access particular applications, perform various operations or manage the delegation and transfer of tasks, as well as when and under what circumstances they are permitted to do so. Authorization privileges depend upon a constantly evolving set of users, applications, partners, and business polices that comprise the enterprise security policy. A typical enterprise environment consists of several thousand users, hundreds of applications, and a myriad of network resources, resulting in a security policy that can consist of tens or hundreds of thousands of interrelated policy rules.

Typically, organizations attempt to control access to the internals of in-house applications through policy rules that are hard-coded in the application or through stored procedure statements in the database. But as the number of applications and databases grows, this patchwork approach to authorization quickly gets out of hand. First, organizations must incur the costly and time-consuming overhead of developing customized security code for each application. But more importantly, once the code is developed and embedded in an application, the embedded policy rules become impossible to track, difficult to update, and nearly impossible to manage because they are scattered throughout the organization.

With an estimated 80 percent of all security breaches coming from authorized users (source: Forrester Research), advanced policy features and enforcement mechanisms are needed to control access to sensitive information assets. To implement an enterprise policy, organizations need a centralized policy and a powerful way to specify policy rules to give them adequate access control security. At the same time, they need a distributed authorization infrastructure to provide authorization services to all applications with performance and scalability for modern distributed network environments.

Therefore, for the foregoing reasons, an improved system and method are needed to protect the distributed networks of enterprises against unauthorized access to their

IPR2024-00027
CrowdStrike Exhibit 1002 Page 601

valuable information assets by managing and enforcing the complex security policy requirements of the organization.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 602

SUMMARY OF THE INVENTION

In accordance with the present invention, a system and method are disclosed to manage and enforce complex security requirements for a computer system in a distributed computer network.

It is therefore an object of the present invention to provide an access control system that can manage individual transactions by users around well-defined, detailed objects within an application. It is also an object of the present invention to provide a policy manager that enables the creation, modification, querying, and analysis of an enterprise access-control policy, as well as the configuration and monitoring of integrated audit logs, while delivering the performance and scalability required to meet the demands of any enterprise. It is a further object of the present invention to provide a system that combines a centrally managed policy database with distributed authorization (access control) services that enforce the policy for all applications across the organization.

It is also an object of this invention to provide a system that works in conjunction with any authentication system, including digital certificates and smartcards, and obviates the need for single sign-on systems by letting organizations set detailed, dynamic rules for exactly who can access which applications, databases, and other network objects. It is a still further object of this invention to provide a robust security policy and authorization service that can be implemented in very heterogeneous environments, across all applications and databases within the organization, thereby completely eliminating the need for embedded, custom security code within applications, and making it possible to centrally manage and administer a consistent, robust security policy for all applications, databases, and network resources. Furthermore, organizations would no longer have to rely on authorization mechanisms provided by packaged or web application vendors that do not integrate with in-house or other third-party products.

In the preferred embodiment, the system comprises a policy manager located on a server for managing and distributing a local client policy based on a global security policy, and an application guard located on a client or server associated with one or more clients for managing access to securable components as specified by the local client policy. The global policy specifies access privileges of the user to securable

IPR2024-00027
CrowdStrike Exhibit 1002 Page 603

components. The policy manager may then distribute a local client policy based on the global policy to the client or server. An application guard located on the client or server then manages authorization requests to the securable components as specified by the local client policy. Each authorization request may be recorded in an audit log to keep track of the authorization requests, whether they were granted or denied, and other useful information.

The system and method of the present invention supports centralized management and distributed authorization. A central policy server stores and manages the policy rules in a centrally administered database. A powerful graphical user interface is used to create, manage, and customize the elements of a policy. Security rules can be specified by both novice and expert users. A dedicated authorization service is associated with one or more applications. The central policy server automatically distributes (over the network) only the relevant portion of the enterprise policy to each remote service. This distributed architecture ensures that authorization requests are not bottlenecked at a central service point and provides unlimited scalability and maximum performance, regardless of the number of applications or policy rules involved.

A more sophisticated security policy is possible because the application has the ability to evaluate access privileges upon every access to the information, during every transaction, and at every data request.

Therefore, the present invention more efficiently and effectively manages and protects computer applications, databases, and network resources against unauthorized access in a distributed computer network.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 604

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one embodiment for one system, in accordance with the present invention;

FIG. 2 is a block diagram of one embodiment of the non-volatile memory located within the server in FIG. 1, according to the present invention;

FIG. 3 is a block diagram of one embodiment of the non-volatile memory located within the client in FIG. 1, according to the present invention;

FIG. 4 is a block diagram of one embodiment of the policy manager located within the non-volatile memory in FIG. 2, in accordance with the present invention;

FIG. 5 is a block diagram of one embodiment of the application guard located within the non-volatile memory in FIG. 3, according to the present invention;

FIG. 6 is a block diagram of one embodiment of a policy loader, in accordance with the present invention;

FIG. 7 is a flowchart of one embodiment of method steps to configure a system, in accordance with the present invention;

FIG. 8 is a flowchart of one embodiment to manage policy in the management station, according to the present invention;

FIG. 9 is a flowchart of one embodiment to navigate tree in the management station, according to the present invention;

FIG. 10 is a flowchart of one embodiment to analyze policy in the management station, in accordance with the present invention;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 605

FIG. 11 is a flowchart of one embodiment to edit policy in the management station, in accordance with the present invention;

FIG. 12 is a flowchart of method steps to distribute policy, according to one embodiment of the present invention;

FIG. 13 is a flowchart of method steps for client access authorization, in accordance with one embodiment of the present invention; and

FIG. 14 is a flowchart of method steps to evaluate authorization request, according to one embodiment of the present invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 606

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to an improvement in security techniques to protect computer systems against unauthorized access. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention includes a system and method for managing and enforcing complex security requirements in a distributed computer network, and comprises a policy manager located on a server for managing and distributing a policy to a client, and an application guard located on the client, the application guard acting to grant or deny access to various components of the client, as specified by the policy.

Referring now to FIG. 1, a block diagram of one embodiment of a distributed computer network system 110 is shown, including a server 112 connected via a network 114 to a client 116, in accordance with the present invention. One client 116 is shown, but server 112 is typically connected to many clients 116. In the FIG. 1 embodiment, server 112 may preferably include a central processing unit (CPU) 118, a read-only memory (ROM) 120, a random-access memory (RAM) 122, a non-volatile memory 124, an input device 126, and a display 128 all connected via a bus 130.

Similarly client 116 may preferably include a central processing unit (CPU) 132, a read-only memory (ROM) 134, a random-access memory (RAM) 136, a non-volatile memory 138, an input device 140, and a display 142 all connected via a bus 144.

Server 112 preferably contains a program stored in non-volatile memory 124 for managing a policy or a set of rules and then distributing the policy to client 116 via link 114. Client 116 preferably contains a program stored in non-volatile memory 138 for granting or denying access to various components or resources of client 116, as specified by the policy distributed from server 112. For example, various components or resources of client 116 can include applications, functions or procedures within an application, data

IPR2024-00027
CrowdStrike Exhibit 1002 Page 607

structures within an application, and database or file system objects referenced by an application.

Referring now to FIG. 2, a block diagram of one embodiment for non-volatile memory 124, located within server 112 of FIG. 1, is shown. In the FIG. 2 embodiment, non-volatile memory 124 includes a policy manager 210 that manages and distributes a policy. A policy is intended to specify the security requirements for applications and database objects. A policy may contain thousands of "security rules" that describe several constraints, including what applications a particular user can access, what objects (operations) within an application a user can access, and how those privileges are constrained by time, geography, or external events. In general, a policy or authorization policy should constrain access to both applications and the operations within them. The policy may be generalized to groups and hierarchies, not just specified for individual users. This would greatly improve manageability and lead to more comprehensible, business-level policies.

An authorization policy preferably consists of four components, including objects, subjects, privileges, and conditions. Objects may be applications, or the operations within an application. Examples of objects include applications or methods, web pages, database tables or files, and menu items in a graphical user interface. The granularity of objects has a direct impact on the level of security achieved. The less information an object contains, it is less likely that a user has access to information not needed to perform his job function. On the other hand, the granularity of objects should be balanced against the ease of security management. The more information an object contains, the fewer objects that have to be protected, and the smaller the policy is.

Objects are preferably organized into an object hierarchy. If an object represents an application, then its children objects might represent the methods with the application. Similarly, if an object represents a database, then its children objects might represent the tables and views within the database.

If a user is granted a certain privilege on a parent object, then he is automatically granted the privilege on all the children objects. Similarly, if a user is denied a certain privilege on a parent object, then he is automatically denied the privilege on all the children objects. In other words, privileges are inherited from parent to children objects. Privilege inheritance through the object hierarchy eases security management because

IPR2024-00027
CrowdStrike Exhibit 1002 Page 608

rather than granting the same privilege to every child object, the privilege is granted once to the parent object, and if the privileges of an object change, the policy on all the children objects automatically reflect the changes made to the object.

Subjects may be users, or roles containing users, who access protected objects. Subjects correspond to users that have access to information in a system. Users can either be internal or external to a system. Users are authorized to access information in order to perform their job functions. Such access may be controlled so that a user gets access only to the information needed to perform his job function.

An object, such as an application or a database, typically has its own list of users. These are users who can log on to the object and be authenticated by the objects, sometimes through an external authentication server. In a large system, users are preferably maintained separately by one or more directory servers. Users are preferably extracted from objects or directory servers, and are maintained up-to-date by synchronizing with these objects and directory servers.

Alias users may also be supported. An alias of a user is another user who inherits all the privileges of the user under certain conditions. Alias facilitates authorization management by providing fine granularity of control on the propagation of privileges. For example, an alias of a user can be created to perform his job function while he is absent. The inheritance of privileges takes effect only when the user is absent. An alias implements the business requirements of delegation, where the privileges of a user can be delegated to another user under certain conditions. Conditional inheritance of privileges through an alias reduces the burden of security management, because it restricts privilege propagation to situations when certain conditions are satisfied.

Users of an object may be defined as being local to that object. In a typical system, the same user is often represented by different login identifications in different objects. This system may support the notion of a "global" user to capture this situation. Every global user is mapped to a set of local users, one per object. Global users facilitate the centralized management of users throughout the system, even if they are identified by different names in different objects.

A privilege defines the kinds of access that may be allowed on objects. In the preferred embodiment, a privilege is the right to perform a particular action on a specific object. The kinds of privileges that apply to an object depend on the type of the object.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 609

Examples of privileges include the right to execute an application, the right to download a web page, the right to query a database table, or the right to view a menu item.

Privileges are granted to users so they can accomplish tasks required for their job. A privilege should be granted to a user only when it is absolutely required for the user to accomplish a task. Excessive granting of unnecessary privileges may lead to compromised security. A user may receive a privilege in two different ways, privileges can be granted to users explicitly (for example, user SMITH can be granted the privilege to execute the payroll application), or privileges can be granted to a role (a named group of privileges), which is then granted to one or more users (for example, a role named "clerk" can be granted the privilege to execute the payroll application, and user SMITH can be granted the clerk role).

Roles are named groups of privileges that are granted to users or other roles. Users granted to a role are the members of that role. A role is often used to represent the set of privileges needed to perform a job function.

The members of a role automatically inherit all the privileges granted or denied to the role. In addition, roles may be organized into a role hierarchy, where parent roles are granted to children roles. If a parent role is granted a privilege, then the children roles are automatically granted the privilege. Similarly, if a role is denied a privilege, then the children roles are automatically denied the privilege.

Roles of an object may be defined as being local to that object. In a typical system, the same role is often represented by different names in different objects. This system may support the notion of a "global" role to capture this situation. Every global role is mapped to a set of local roles, one per object. Global roles facilitate the centralized management of roles throughout the system, even if they are identified by different names in different objects.

Role membership may be further constrained by the notion of mutual exclusion. Two roles are mutually exclusive if no single user can be granted to both roles simultaneously. Role mutual exclusion implements a business requirement of separation of duty. For example, a submit_budget role and an approve_budget role should be mutually exclusive, because no user should be simultaneously authorized to perform both actions.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 610

In a typical policy, there are preferably two types of access rules, a grant rule, and a deny rule. A grant rule states that a privilege on an object is granted to a subject under an optional constraint. A deny rule states that a privilege on an object is denied to a subject under an optional constraint. Additionally, a wild card "any" may be used as a privilege, object, or subject, meaning that any legitimate value could be substituted in its place.

An access request preferably consists of a privilege, an object, and a subject, representing the fact that the subject request authorization of the privilege on the object. An access request matches a grant rule if the privilege, object, and subject math those in the rule, and the constraint in the rule evaluates to "true." An access request matches a deny rule if the privilege, object, and subject match those in the rule, and the constraint in the rule does not evaluate to "false."

An access request is denied if there is a deny rule matching the request, or there are no access rules matching the request. An access request is granted if there are no deny rules matching the request, and there is a grant rule matching the request.

Conditions define the constraints on when objects and subjects can be accessed. The constraints in an access rule specifies further requirements on when the access rule is applicable. These requirements could be conditioned on properties of the object or the subject.

Constraints are preferably expressions formed from conditions and Boolean operators NOT, AND, and OR. Three kinds of built-in conditions may be used: 1) relational operations =, <>, <, <=, >, >= on integers; 2) relational operations =, <>, LIKE, NOTLIKE on strings (the operator LIKE takes a string and a pattern and evaluates to true if the string matches the pattern, the operator NOTLIKE is the negation of LIKE); and 3) set operations IN, NOTIN (the operator IN on integers takes an integer and a set of integers and evaluates to "true" if the integer is in the set, the operator IN on strings is similarly defined, and the operator NOTIN is the negation of IN).

In addition to built-in conditions, users of system 110 may declare custom evaluation functions, which are customer-defined conditions. System 110 may provide an Application Programming Interface (API) for invoking customer-supplied code to evaluate custom evaluation functions. For example, an evaluation function could access

IPR2024-00027
CrowdStrike Exhibit 1002 Page 611

a remote database to validate certain properties of the object. Another evaluation function could invoke an external server to authenticate the subject.

Now referring to the FIG. 2 embodiment, policy manager 210 preferably includes a management station program 212 to operate policy manager 210, a distributor program 214 to distribute local client policies to clients, a logger program 216 to track authorization requests, and a database management system (DBMS) 218 to maintain policy data files. Policy manager 210 also includes an audit log data file 220 to record authorization requests, an optimized policy data file 222, an enterprise policy data file 224, an administrative policy data file 226, and a local administrative policy data file 228. The contents and operation of policy manager 210 are further discussed below in conjunction with FIGS. 4, 8, 9, 10, 11, and 12.

Referring now to FIG. 3, a block diagram of one embodiment for non-volatile memory 138, located within client 116 of FIG. 1, is shown. In the FIG. 3 embodiment, non-volatile memory 138 preferably includes an application guard 310 that grants or denies access to various components of client 116, as specified by a pre-determined policy. For example, various components of client 116 can include applications, data, and/or objects. In the FIG. 3 embodiment, application guard 310 preferably includes at least one application 312, an authorization library program 314, an authorization engine program 316, and a local client policy 318. The contents and operation of application guard 310 are further discussed below in conjunction with FIGS. 5, 13, and 14.

Referring now to FIG. 4, a block diagram of one embodiment for policy manager 210, located within non-volatile memory 124 in FIG. 2, is shown. In the preferred embodiment, policy manager 210 allows system users to implement, analyze, edit and update a centrally-managed security policy or enterprise policy 224. In the FIG. 4 embodiment, policy manager 210 preferably includes a management console or management station 212, a database management system 218, an audit facility or logger 216, and a distributor 214.

In the FIG. 4 embodiment, management station 212 preferably includes a graphical user interface (GUI) 410 for creating or customizing rules by system users. Management station 212 supports concurrent rule development by multiple users. Each policy rule preferably includes four basic components: 1) an object that is to be protected; 2) an access right or privilege; 3) a global or local user to which the privilege

IPR2024-00027
CrowdStrike Exhibit 1002 Page 612

applies; and 4) conditions under which the privilege is granted or denied, including built-in access criteria such as time of day or location, as well as custom-defined access criteria.

Graphical user interface (GUI) 410 provides a user-friendly set of menu options or management services 412 to fully operate the policy manager. Programs controlled by the menu options may include navigation 414, search 416, distribution 418, edit 420, query 422, and log viewer 424. The operation of these programs are further discussed below in conjunction with FIGS. 8, 9, 10, 11, and 12. As an alternative to the GUI, the management services can be operated from an application through an API that allows programs to perform the same functions as a human operator. In the preferred embodiment, management station also includes an application guard 426 to allow only authorized administrators to operate management station 212. Local administrative policy 228 provides a set of policy rules specifying which users are authorized to access management station 212.

After the policy rules are created or modified using management station 212, they may then be distributed to appropriate clients 116. Management station 212 includes a communication interface 434 in order to pass information between various other components in system 110.

Prior to when the policy rules are distributed, a parser/type checker 428 preferably reviews and reconstructs the policy rules to make sure that they are syntactically and semantically correct according to a predefined policy language. The policy rules pass through a database layer (DB layer) 430 and an open database connectivity layer (ODBC) 432 before being stored as enterprise policy 224. DB layer 430 formats the policy rules into standard database storage tables, and ODBC 432 provides a common interface to various vendor-specific databases.

Enterprise policy 224 is then passed to distributor 214. An optimizer program 436 within distributor 214 determines which application guard 310 needs to receive which policy rules. A differ program 438 determines what type of changes were made to optimized policy 222, and then distributes only the changed policy rules or local client policy 318 to the appropriate application guards 310 through an ODBC layer 440 and a communication interface 442, which enforce access control to local applications and data.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 613

Since the application guards 310 can be distributed among various clients 116, and each application guard 310 has its own specific local client policy 318, the system provides scalability.

Distributor 214 may also be used to optimize administrative policy 226 into an optimized administrative policy or local administrative policy 228 for use with application guard 426 in management station 212.

Referring now to FIG. 5, a block diagram of one embodiment of application guard 310, located within non-volatile memory 138 in FIG. 3, is shown. Application guard 310 may be distributed on clients 116 throughout an enterprise, and is designed to reside along with each of the protected applications having an associated application guard 310.

Application guard 310 supports transactional access control by allowing an application to be aware of the authorization service and to make authorization requests at each and every user interaction, data request, or business-level transaction. In addition, the design and integration of application guard 310 is fundamental to providing access control to business-level objects within an application since the authorization services have visibility to those named policy objects within the application.

In the FIG. 5 embodiment, application guard 310 is preferably integrated into application 312 through a high-level application programming interface (API) or authorization library 314 that allows application 312 to request authorization services as needed through an application guard interface 512. Typically, this can be done very quickly by including authorization requests at key points in application 312 for control user interaction or database access so that each interaction is protected with a minimum of development.

In the FIG. 5 embodiment, an authorization request is processed by authorization engine 316. A parser/type checker 514 parses local client policy 318 and stores the parsed local client policy in RAM 136. An evaluator 516 then determines whether the authorization request should be granted or denied by evaluating the authorization request with the parsed local client policy in RAM 136. Plug-ins 522 in authorization engine 316 allow for additional capabilities to process and evaluate authorization requests based on customized code. Each authorization request is then recorded in an audit log 518 and transmitted to logger 216 via a communication interface 520.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 614

Users have the option of implementing application guard 310 locally to application 312, as a service running on the same system as application 312, or as a remote authorization service through a remote procedure call to another server. The advantage of the latter design would be to offload the application server from handling authorization services or allowing a single authorization server to handle a multiple number of applications. A local implementation would provide maximum performance and minimize any network traffic overhead.

As seen in FIG. 5, application guard 310 includes an application guard interface 512 coupled to an application 312 for requesting access to securable components. Application guard 310 also includes at least one authorization engine 316 for evaluating requests from application guard interface 512 as specified by local client policy 318. Multiple authorization engines 316 can be used for added performance and reliability. Furthermore, application guard interface 512 can be located on a client computer, while authorization engine 316 and local client policy 318 can be located on a client server.

The application guard authorization service of the present invention introduces virtually no performance overhead to an existing application 312. The policy rules developed at policy manager 210 are compiled into an optimized form before being distributed to the target application guards 310. This optimized form only distributes attributes relevant to that application guard 310, so that access requests may be evaluated by reviewing only a few rules rather than frequently analyzing the potentially large policy rule base.

Referring back to FIG. 4, logger 216 may then advantageously receive a client audit log 450 through a communication interface 452 and an ODBC 454 from authorization engine 316 (FIG. 5). Client audit log 450 is then formatted by message processing 456 before being stored in audit log 220. Audit log 220 may then be monitored via log viewer 424 in management station 212.

Referring now to FIG. 6, a block diagram of one embodiment of a policy loader 610 is shown. In the FIG. 6 embodiment, policy rules may be entered one at a time into enterprise policy database 224 via management station 212 (FIG. 4), or the policy rules may alternatively be loaded as a batch process via policy loader 610. Policy loader 610 is an additional utility that bulk loads an existing set of policy rules into enterprise policy database 224. An existing set of policy rules may be entered into policy loader 610 via

IPR2024-00027
CrowdStrike Exhibit 1002 Page 615

input 612. A parser/type checker 614 then preferably reviews and reconstructs the policy rules to make sure that they are syntactically and semantically correct according to a predefined policy language. The policy rules may then be passed through a DB layer 616 and an ODBC 618 before being stored in enterprise policy database 224.

Referring now to FIG. 7, a flowchart of method steps to configure a system in accordance with one embodiment of the present invention is shown. Initially, in step 710, a system administrator installs policy manager 210 on a server 112. The installation may include management station 212, distributor 214, logger 216, and DBMS 218. After all the components of policy manager 210 have been installed, the system administrator then enters a set of policy rules. In step 712, the administrator can decide whether to use policy loader 610, or to use management station 212 to enter policy rules. If the system administrator decides to use management station 212, then at step 714, policy rules are entered using edit function 420. However, if policy loader 610 is used, then at step 716, policy rules are entered into a file, and at step 718, the file of policy rules passes to policy loader 610.

Next, in step 720, the system administrator installs application guards 310 onto client systems 116, as well as installing local client policies 318 onto client systems 116. Then at step 722, the system administrator registers plug-ins 522 into application guards 318 to allow for additional capabilities in order to process authorization requests based on customized code.

Referring now to FIG. 8, a flowchart of one embodiment to manage policy under management services 412 in management station 212 is shown. In order to allow for a complex set of policy rules, a number of different functionality features are incorporated into system 110 to facilitate implementation and management.

In the FIG. 8 embodiment, at step 810, an authorized administrator logs in to policy manager 210. Next, in step 812, the authorized administrator chooses between administrative mode or enterprise mode. The administrative mode allows the system administrator to manage administrative policy 226, and the enterprise mode allows the system administrator to manage enterprise policy 224. The system administrator is then presented with six menu options including navigate tree 814, analyze policy 816, edit policy 818, distribute policy 820, view audit log 822, and exit 824. The features of navigate tree 814, analyze policy 816, edit policy 818, and distribute policy 820 are

IPR2024-00027
CrowdStrike Exhibit 1002 Page 616

described in more detail through FIGS. 9, 10, 11, and 12, respectively. View audit log 822 is a security feature that allows an administrator to view and track all authorization requests that have occurred at any application guards 310 connected to system 110. The system administrator can choose any of the menu options, or at step 824 the system administrator may then exit the system.

Referring now to FIG. 9, a flowchart of one embodiment of menu option navigate tree 814 in management station 212 is shown. Navigate tree 814 provides a set of options for an administrator to add, delete, and/or modify features on server 112 or client 116. The features that an administrator may add, delete, and/or modify include global users 910, global roles 912, directories 914, local roles 916, local users 918, applications 920, application guards 922, and declarations 924. At step 926, the system administrator may then exit from navigate tree 814.

Referring now to FIG. 10, a flowchart of one embodiment of menu option analyze policy 816 in management station 212 is shown. Analyze policy 816 preferably allows an authorized user to analyze and view rules and policies within enterprise policy 224. At step 1010, the user has an option to search rules, or at step 1012 to query policy. When search rules is selected, a search can be made for all the grant rules or all the deny rules pertaining to a particular user. When query policy is selected, a search can be made on who is granted or denied what privilege on which objects under what conditions.

After analyzing and viewing rules or policies, at step 1014 the system administrator may exit from analyze policy 816.

Referring now to FIG. 11, a flowchart of one embodiment of menu option edit policy 818 in management station 212 is shown. Edit policy 818 allows an authorized user to add, delete, and/or modify enterprise policy 224 features. The features that may be edited include rule sets 1110, access 1112, privilege 1114, objects 1116, user/role 1118, and attributes 1120. At step 1122, the system administrator may then exit edit policy 818.

Referring now to FIG. 12, a flowchart of one embodiment of method steps of menu option distribute policy 820 is shown. After enterprise policy 224 has been initially entered or modified in any way, the modified features of enterprise policy 224 may then be distributed to appropriate application guards 310. At step 1210, upon selecting the distribute policy option, distributor 214 optimizes enterprise policy 224.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 617

Then at step 1212, differ 438 preferably computes any difference between the newly optimized policy and optimized policy 222. At step 1214, the newly optimized policy is then published as optimized policy 222 in DBMS 218. Next, at step 1216, only the changed portions of optimized policy 222 are committed to appropriate application guards 310. At step 1218, application guards 310 receive the changed policy, and then at step 1220, application guards 310 merge the changed policy into local client policy 318. Next at step 1222, new local client policy 318 is activated to work with application guard 310.

Referring now to FIG. 13, a flowchart of one embodiment of method steps for client access authorization is shown. The FIG. 13 example for using a standard application guard 310 by a user begins with a user requesting access to a securable component protected by an application guard 310. In step 1310, application guard 310 constructs and issues an authorization request. At step 1312, the authorization request is evaluated by application guard 310 according to its local client policy 318 to determine whether to allow or deny the authorization request. At step 1314, audit 518 then records the authorization request in audit log 450. Next, at step 1316, if there is an error in the authorization request, or if the request is not valid, then at step 1318 the user is denied access. However, if the authorization request is valid, then at step 1320 it is determined whether access should be granted. If the evaluated authorization request does not deny access for the user, then at step 1322 access is allowed. If the evaluated authorization request denies access for the user, then at step 1324 access is denied.

Referring now to FIG. 14, a flowchart of one embodiment of method steps to evaluate an authorization request from an application guard 310 is shown. In order to evaluate an authorization request at application guard 310, in step 1420, evaluator 516 first searches any deny rules in local policy 318. At step 1412, if evaluator 516 finds any deny rules, then at step 1414, an evaluation is performed on any constraints on the deny rules. If, at step 1416, the evaluation finds presently valid constraints on the deny rules, then at step 1418 access is denied. However, if at step 1416, the evaluation finds that the constraints on the deny rules are not presently valid, or if no deny rules are found in foregoing step 1412, then at step 1420, a search of grant rules is performed. If no grant rules are found at step 1422 that would allow access for the user, then at step 1418, access is denied. If in step 1422 grant rules are found, then at step 1424 an evaluation is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 618

performed on any constraints in the grant rules. If the evaluated constraint is presently valid, then at step 1426, a true value is passed, and at step 1428 access is allowed. However, if the evaluated constraint is not presently valid, then at step 1426, a false value is passed, and at step 1418 access is denied.

The invention has been explained above with reference to a preferred embodiment. Other embodiments will be apparent to those skilled in the art in light of this disclosure. For example, the present invention may readily be implemented using configurations other than those described in the preferred embodiment above. Additionally, the present invention may effectively be used in conjunction with systems other than the one described above as the preferred embodiment. Therefore, these and other variations upon the preferred embodiments are intended to be covered by the present invention, which is limited only by the appended claims.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 619

WHAT IS CLAIMED IS:

1. A system for maintaining security in a distributed computing environment, comprising:

a policy manager for managing a security policy; and

an application guard for managing access to securable components as specified by the security policy.

2. The system of claim 1, wherein said policy manager comprises a management station for constructing and editing the security policy.

3. The system of claim 2, wherein said policy manager further comprises a distributor for distributing the security policy to a client.

4. The system of claim 2, wherein said policy manager further comprises a distributor for distributing a customized local policy based on the security policy to a client.

5. The system of claim 4, wherein said policy manager further comprises a logger for recording and tracking authorization events that occur through the application guard.

6. The system of claim 4, wherein the policy manager further comprises a database management system for maintaining the security policy.

7. The system of claim 4, wherein the customized local policy is optimized.

8. The system of claim 1, wherein said securable components are selected from the group consisting of: at least one application, a function within an application, a procedure within an application, a data structure within an application, a database object referenced by an application, or a file system object referenced by an application.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 620

9. The system of claim 1, wherein said system is scalable by further comprising a plurality of clients, said policy manager further managing and distributing a customized local policy to each client, and at least one additional application guard located on each client for managing access to the securable components as specified by each customized local policy.

10. The system of claim 1, wherein said application guard includes an application guard interface coupled to an application for requesting access to the securable components, and at least one authorization engine for evaluating requests from the application guard interface as specified by a customized local policy based on the security policy.

11. The system of claim 10, wherein said application guard interface is located on a client, and said at least one authorization engine and said customized local policy are located on a client server.

12. The system of claim 1, wherein the security policy is defined by a policy language to grant or deny access to the securable components for a particular user.

13. The system of claim 1 further comprising a policy loader for bulk loading the security policy onto the system.

14. The system of claim 1, wherein said policy manager includes a set of menu options to manage and distribute the security policy.

15. The system of claim 14, wherein said set of menu options include: navigate tree, analyze policy, edit policy, distribute policy, and view audit log.

16. The system of claim 1, wherein the application guard further allows for additional customized code to process and evaluate authorization requests based on the additional customized code.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 621

17. The system of claim 1, wherein the policy manager further includes a policy manager application guard for managing access to the policy manager as specified by a local administrative policy.

18. A system for controlling user access in a distributed computing environment, comprising:

a global policy specifying access privileges of the user to securable components;

a policy manager located on a server for managing and distributing a local client policy based on the global policy to a client, and

an application guard located on the client for managing access to the securable components as specified by the local client policy.

19. The system of claim 18 further comprising at least one additional client, said policy manager further managing and distributing a customized local policy based on the global policy to each additional client, and at least one additional application guard located on each additional client for managing access to the securable components as specified by the customized local policy.

20. The system of claim 18, wherein said policy manager comprises:

a management station for constructing and editing the global policy; and

a distributor for distributing the local policy to the client.

21. The system of claim 20, wherein said policy manager further comprises a database management system for maintaining the global policy.

22. The system of claim 18, wherein said policy manager further comprises a logger for recording and tracking authorization events that occur through the application guard.

23. The system of claim 18, wherein said securable components are selected from the group consisting of: at least one application, a function within an application, a procedure within an application, a data structure within an application, a database object referenced by an application, or a file system object referenced by an application.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 622

24. The system of claim 18, wherein the global policy is defined by a policy language to grant or deny access to the securable components for a particular user.

25. The system of claim 18, wherein said system is scalable by further comprising a plurality of clients, said policy manager further managing and distributing a customized local policy to each client, and at least one additional application guard located on each client for managing access to the securable components as specified by each customized local policy.

26. The system of claim 18, wherein said application guard includes an application guard interface coupled to an application for requesting access to the securable components, and at least one authorization engine for evaluating requests from the application guard interface as specified by the local client policy.

27. The system of claim 18 further comprising a policy bulk loader for bulk loading the global policy onto the system.

28. The system of claim 18, wherein the local client policy is optimized.

29. The system of claim 18, wherein the application guard further allows for additional customized code to process and evaluate authorization requests based on the additional customized code.

30. The system of claim 18, wherein the policy manager further includes a policy manager application guard for managing access to the policy manager as specified by a local administrative policy.

31. A system for authorization that provides access to securable components for a user, comprising:

a policy specifying access privileges of the user to the securable components;

an application guard; and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 623

a processor coupled to said system, said processor executing said application guard to manage access to the securable components.

32. The system of claim 31 further including an audit log for recording and tracking each authorization event that occurs though the application guard.

33. The system of claim 31, wherein said securable components are selected from the group consisting of: at least one application, a function within an application, a procedure within an application, a data structure within an application, a database object referenced by an application, or a file system object referenced by an application.

34. The system of claim 31, wherein said application guard comprises an application guard interface connected to an application for managing access to the securable components in said application.

35. The system of claim 31, wherein the application guard further allows for additional customized code to process and evaluate authorization requests based on the additional customized code.

36. The system of claim 31, wherein said application guard includes an application guard interface coupled to an application for requesting access to the securable components, and at least one authorization engine for evaluating requests from the application guard interface as specified by a customized local policy based on the policy.

37. The system of claim 36, wherein said application guard interface is located on a client, and said at least one authorization engine and said customized local policy are located on a client server.

38. A system for managing security in a distributed computing environment, comprising:

a policy manager; and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 624

a processor coupled to said system, said processor executing said policy manager to manage and distribute a customized local policy based on a global policy to a client.

39. The system of claim 38, wherein said policy manager comprises a management station for constructing and editing the global policy.

40. The system of claim 39, wherein said policy manager further comprises a distributor for distributing the customized local policy to the client.

41. The system of claim 38, wherein said policy manager further comprises a logger for recording and tracking authorization events that are received from the client.

42. The system of claim 38, wherein said global policy specifies access privileges of at least one user to securable components.

43. The system of claim 38, wherein the policy manager comprises a database management system for maintaining the global policy.

44. The system of claim 38, wherein the global policy is defined by a policy language to grant or deny access to securable components for a particular user.

45. The system of claim 38, wherein said policy manager includes a set of menu options to manage and distribute the customized local policy.

46. The system of claim 38, wherein the customized local policy is optimized.

47. The system of claim 38, wherein the policy manager further includes a policy manager application guard for managing access to the policy manager as specified by a local administrative policy.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 625

48. A method for maintaining security in a distributed computing environment, comprising the steps of:

managing a policy using a policy manager by specifying access privileges of a user to securable components; and

distributing the policy to a client having an application guard, whereby the application guard manages access to the securable components as specified by the policy.

49. The method of claim 48, further including the step of recording authorization events that occur through the application guard after distributing the policy.

50. The method of claim 48, wherein the securable components are selected from the group consisting of: at least one application, a function within an application, a procedure within an application, a data structure within an application, a database object referenced by an application, or a file system object referenced by an application.

51. A method for maintaining security on a client in a distributed computing environment, comprising the steps of:

constructing and issuing an authorization request for a user to access to securable components located on the client using an application guard;

evaluating the authorization request using the application guard to determine if the authorization request is valid or invalid; and

allowing access to the user via the application guard if the evaluated authorization request was valid, and denying access to the user via the application guard if the authorization request was invalid.

52. The method of claim 51, after evaluating the authorization request, further including the step of recording the authorization request in an audit log.

53. A computer-readable medium comprising program instructions for maintaining security in a distributed computing environment by performing the steps of:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 626

managing a policy using a policy manager by specifying access privileges of a user to securable components;

distributing the policy using the policy manager to a client having an application guard, whereby the application guard manages access to the securable components as specified by the policy; and

executing said policy manager with a processor to manage and distribute the policy.

54. A computer-readable medium comprising program instructions for maintaining security on a client in a distributed computing environment by performing the steps of:

constructing and issuing an authorization request for a user to access to securable components located on the client using an application guard;

evaluating the authorization request using the application guard to determine if the authorization request is valid or invalid;

allowing access to the user via the application guard if the evaluated authorization request was valid, and denying access to the user via the application guard if the authorization request was invalid; and

executing said application guard with a processor to automatically maintain security on the client.

55. A system for maintaining security in a distributed computing environment, comprising:

means for managing a policy using a policy manager by specifying access privileges of a user to securable components;

means for distributing the policy using the policy manager to a client having an application guard, whereby the application guard manages access to the securable components as specified by the policy; and

means for executing the policy manager to manage and distribute the policy.

56. A system for maintaining security on a client in a distributed computing environment, comprising:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 627

means for constructing and issuing an authorization request for a user to access to securable components located on the client using an application guard;

means for evaluating the authorization request using the application guard to determine if the authorization request is valid or invalid;

means for allowing access to the user via the application guard if the evaluated authorization request was valid, and denying access to the user via the application guard if the authorization request was invalid; and

means for executing said application guard to automatically maintain security on the client.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 628

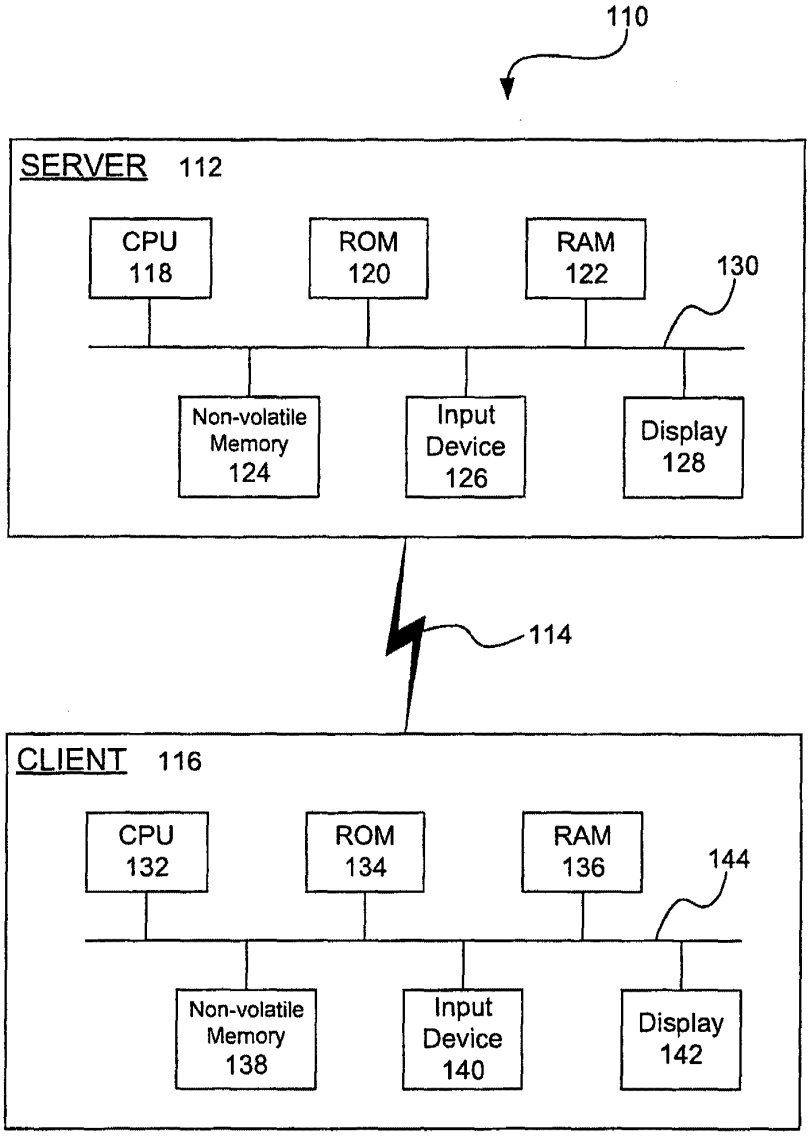

110
SERVER 112
CPU 118
ROM 120
RAM 122
130
Non-volatile Memory 124
Input Device 126
Display 128
114
CLIENT 116
CPU 132
ROM 134
RAM 136
144
Non-volatile Memory 138
Input Device 140
Display 142

FIG. 1

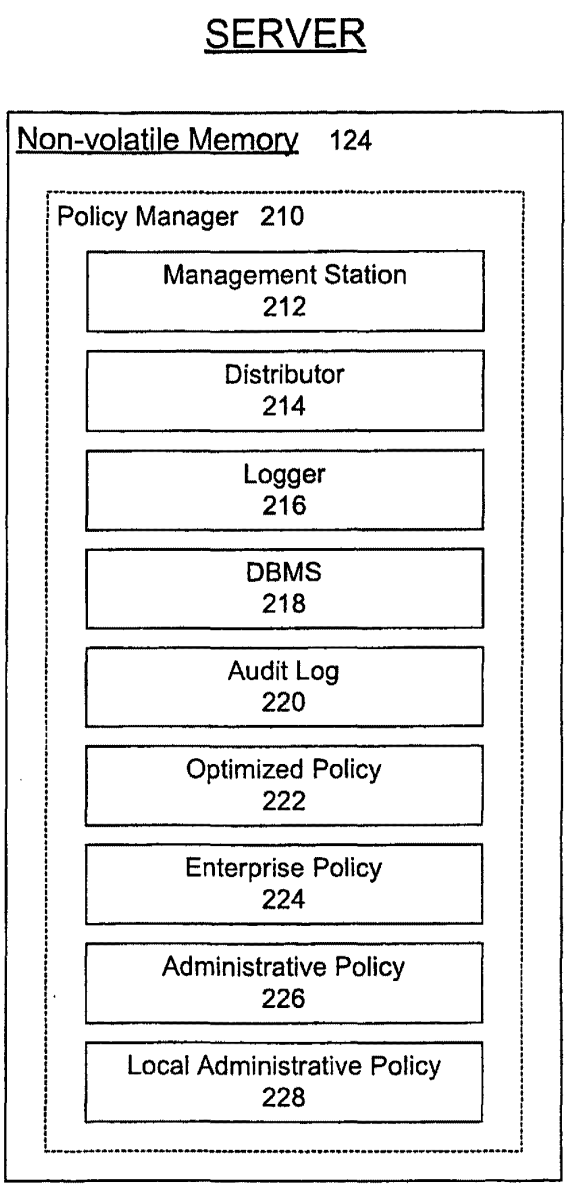

SERVER
Non-volatile Memory 124
Policy Manager 210
Management Station 212
Distributor 214
Logger 216
DBMS 218
Audit Log 220
Optimized Policy 222
Enterprise Policy 224
Administrative Policy 226
Local Administrative Policy 228

FIG. 2

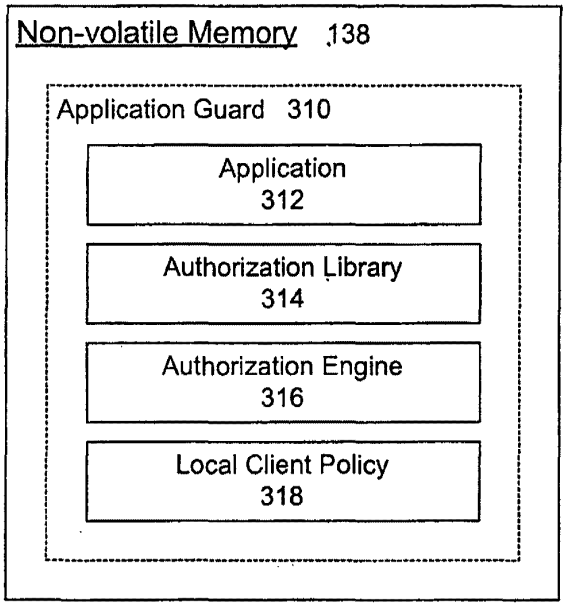

CLIENT
Non-volatile Memory 138
Application Guard 310
Application 312
Authorization Library 314
Authorization Engine 316
Local Client Policy 318

FIG. 3

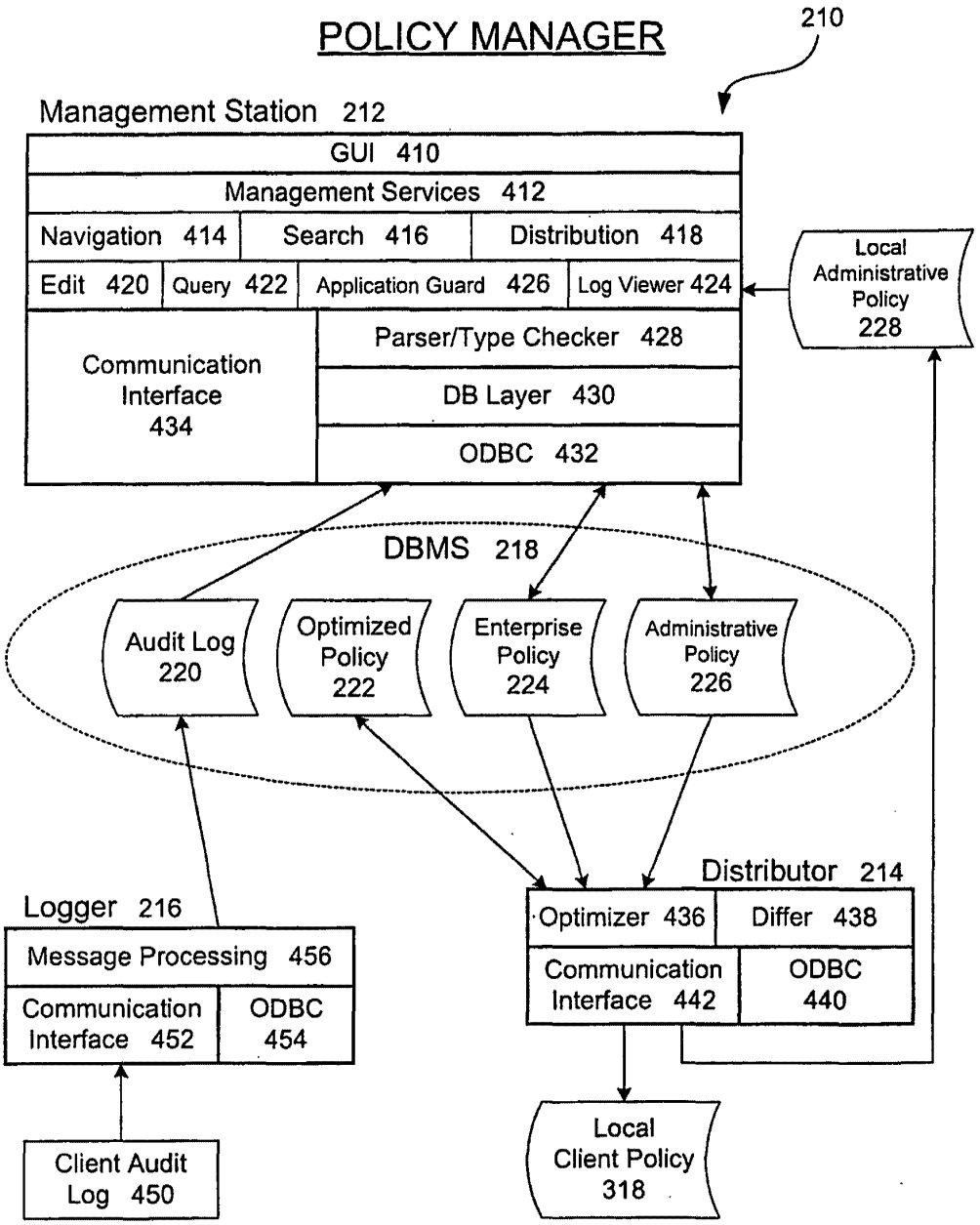

POLICY MANAGER
210
Management Station 212
GUI 410
Management Services 412
Navigation 414
Search 416
Distribution 418
Edit 420
Query 422
Application Guard 426
Log Viewer 424
Communication Interface 434
Parser/Type Checker 428
DB Layer 430
ODBC 432
Local Administrative Policy 228
DBMS 218
Audit Log 220
Optimized Policy 222
Enterprise Policy 224
Administrative Policy 226
Distributor 214
Optimizer 436
Differ 438
Communication Interface 442
ODBC 440
Logger 216
Message Processing 456
Communication Interface 452
ODBC 454
Client Audit Log 450
Local Client Policy 318

FIG. 4

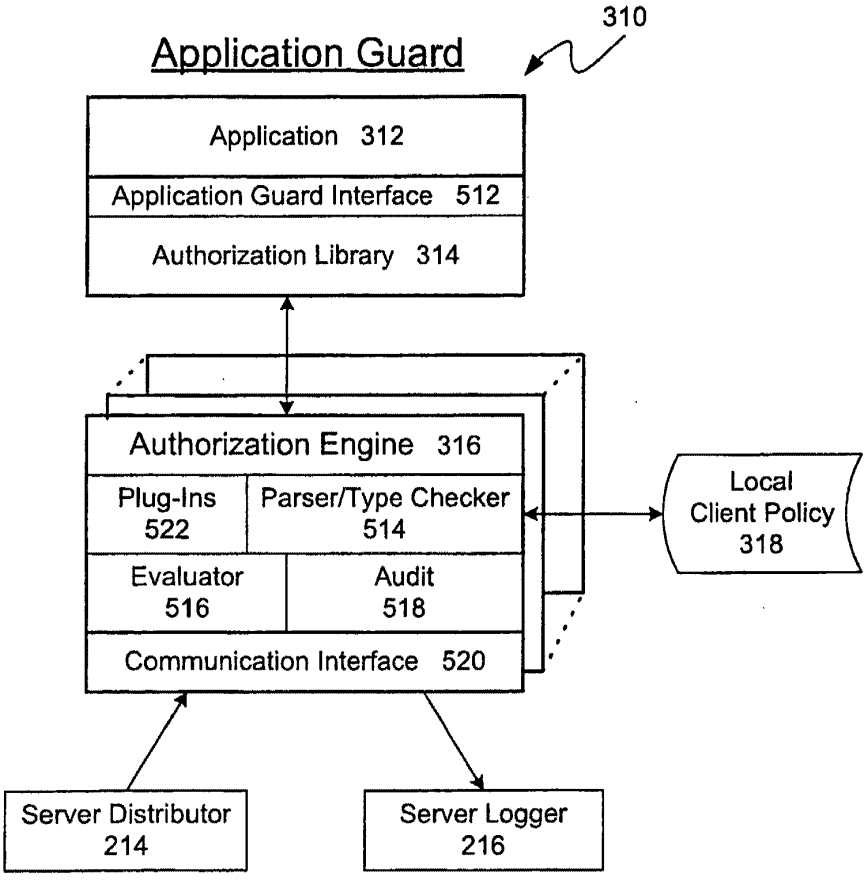

310
Application Guard
Application 312
Application Guard Interface 512
Authorization Library 314
Authorization Engine 316
Plug-Ins 522
Parser/Type Checker 514
Evaluator 516
Audit 518
Communication Interface 520
Local Client Policy 318
Server Distributor 214
Server Logger 216

FIG. 5

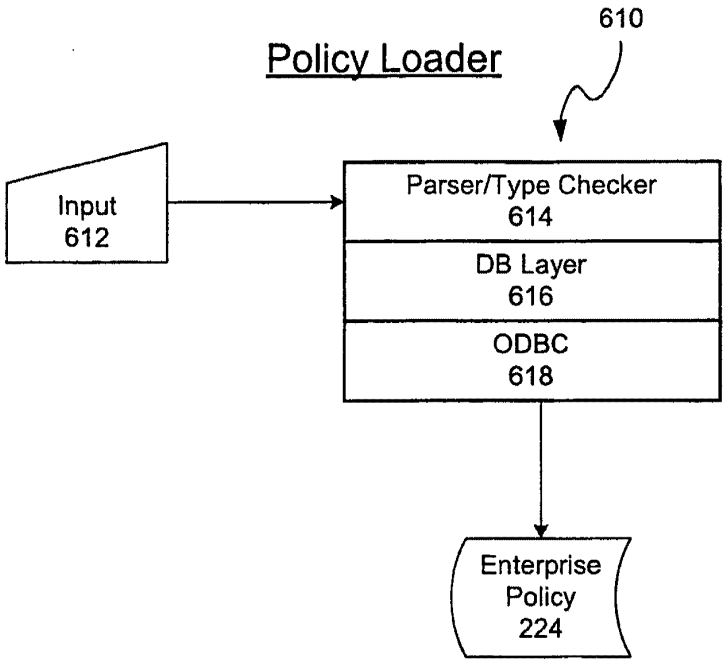

610
Policy Loader
Input
612
Parser/Type Checker
614
DB Layer
616
ODBC
618
Enterprise
Policy
224

FIG. 6

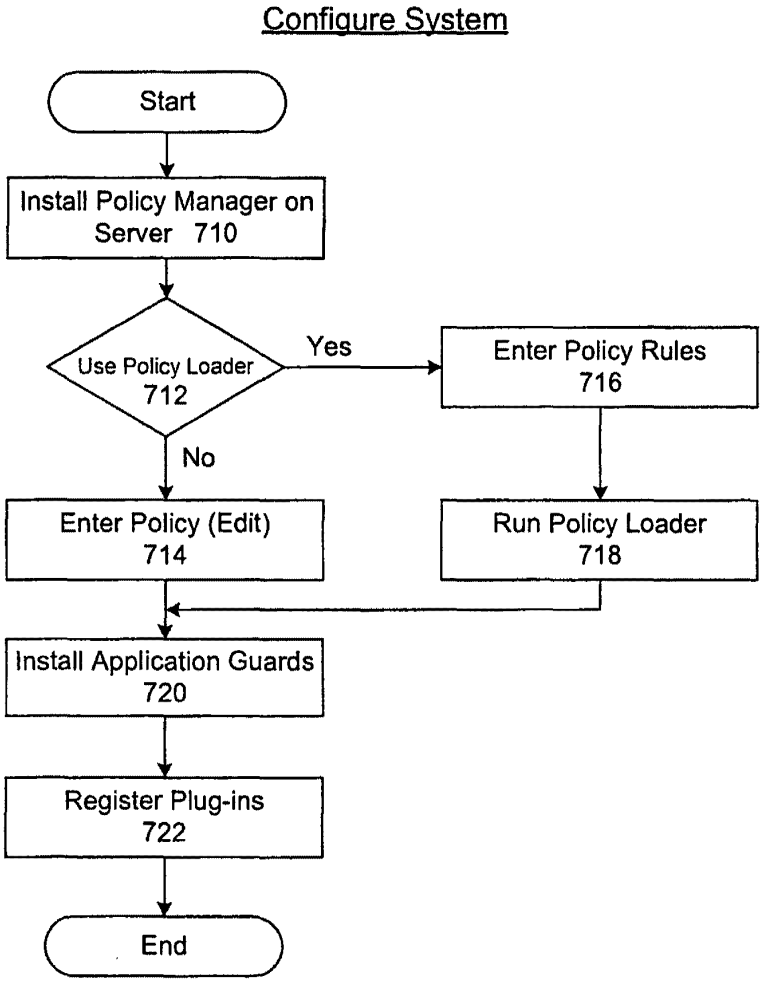

Configure System
Start
Install Policy Manager on Server 710
Use Policy Loader 712
Yes
Enter Policy Rules 716
No
Enter Policy (Edit) 714
Run Policy Loader 718
Install Application Guards 720
Register Plug-ins 722
End

FIG. 7

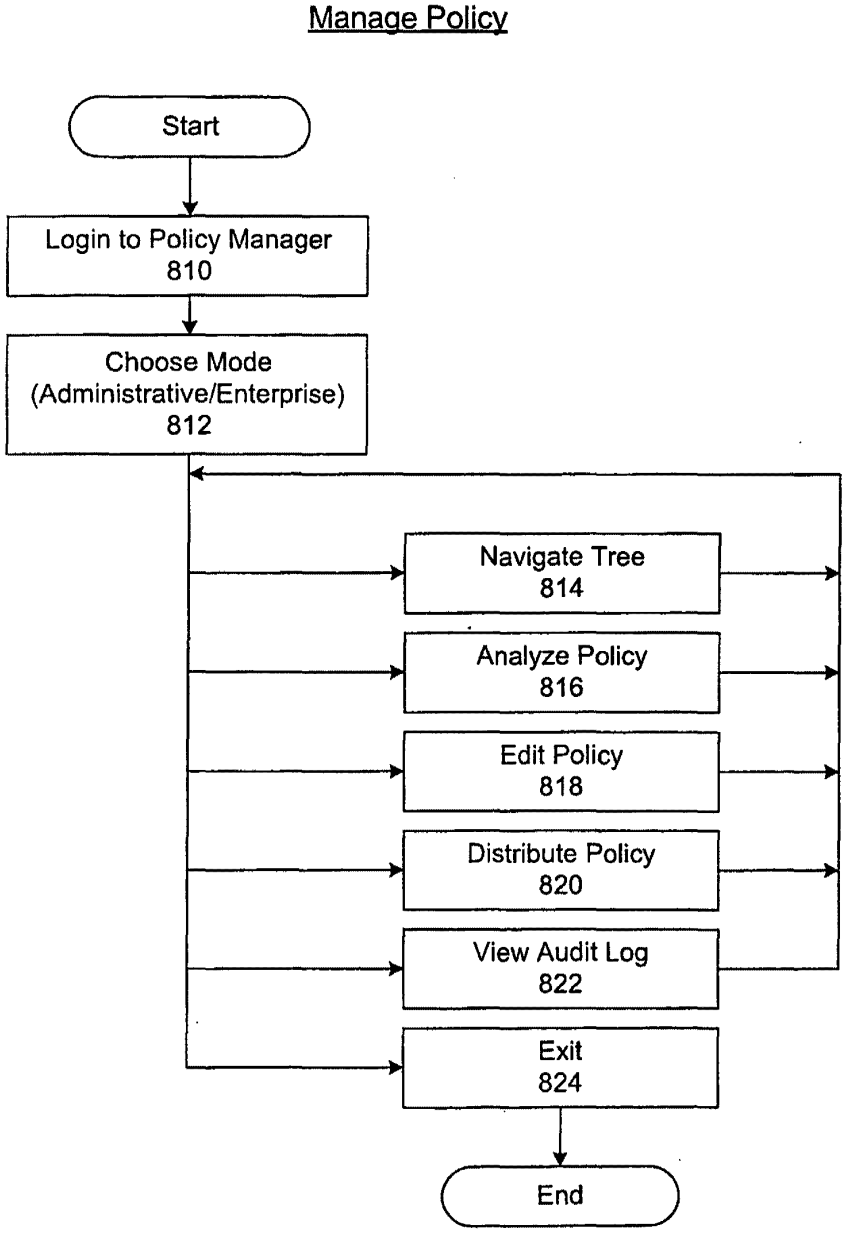

Manage Policy
Start
Login to Policy Manager
810
Choose Mode
(Administrative/Enterprise)
812
Navigate Tree
814
Analyze Policy
816
Edit Policy
818
Distribute Policy
820
View Audit Log
822
Exit
824
End

FIG. 8

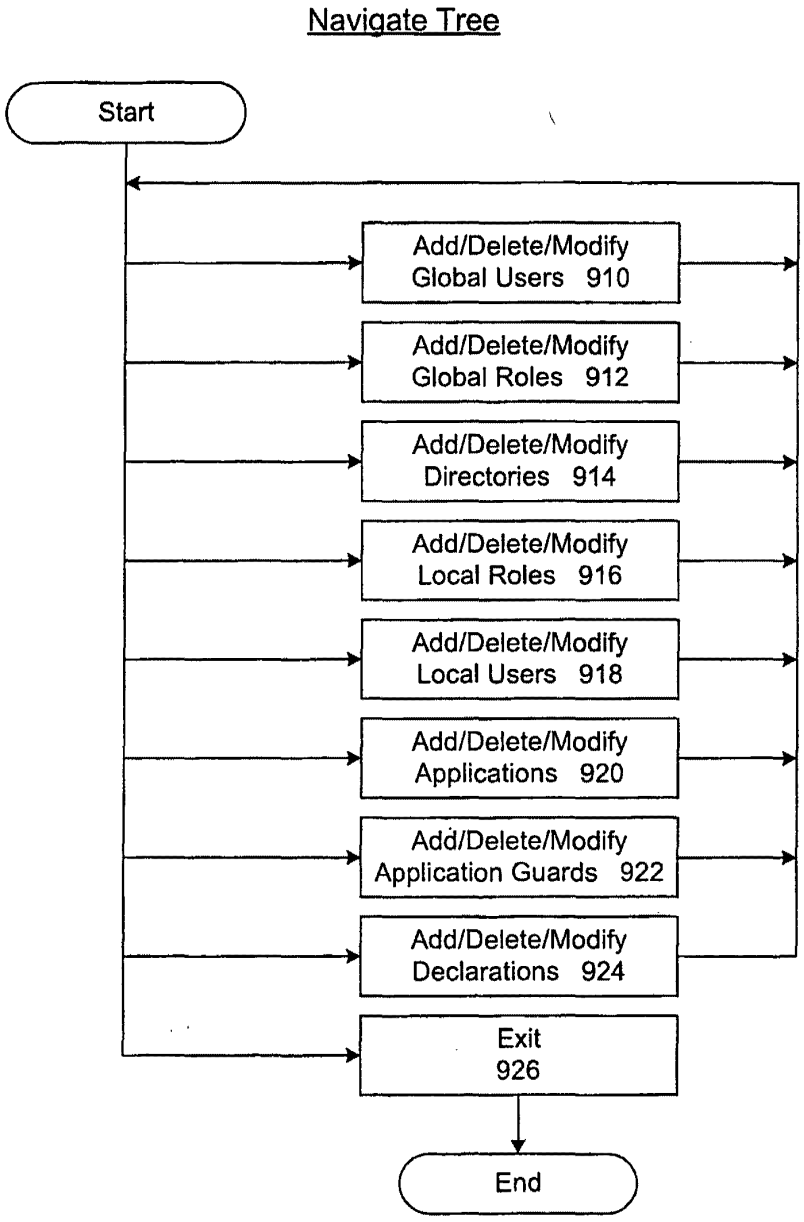

Navigate Tree
Start
Add/Delete/Modify Global Users 910
Add/Delete/Modify Global Roles 912
Add/Delete/Modify Directories 914
Add/Delete/Modify Local Roles 916
Add/Delete/Modify Local Users 918
Add/Delete/Modify Applications 920
Add/Delete/Modify Application Guards 922
Add/Delete/Modify Declarations 924
Exit 926
End

FIG. 9

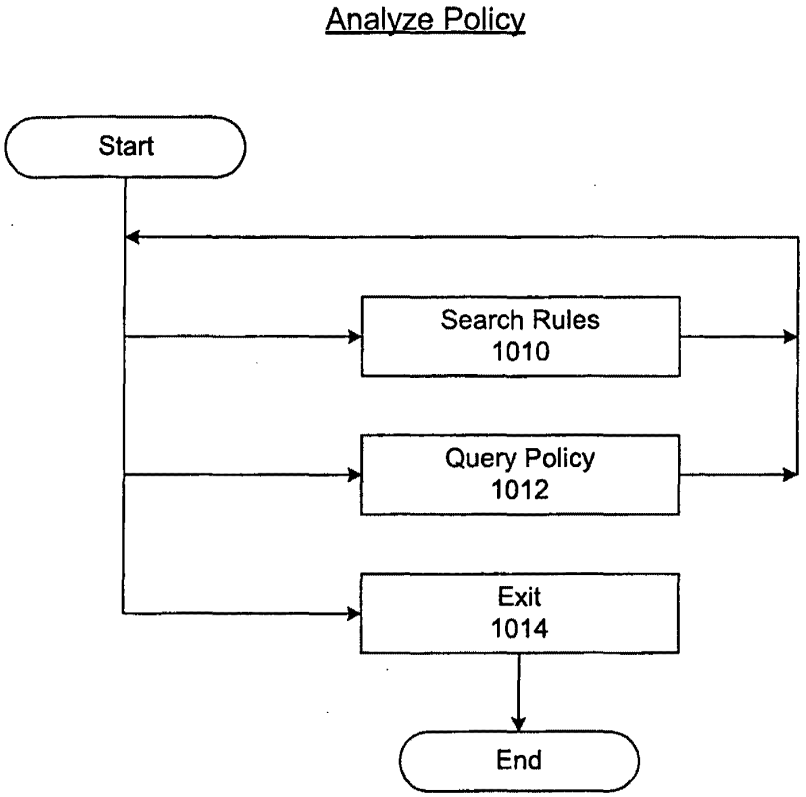

Analyze Policy
Start
Search Rules
1010
Query Policy
1012
Exit
1014
End

FIG. 10

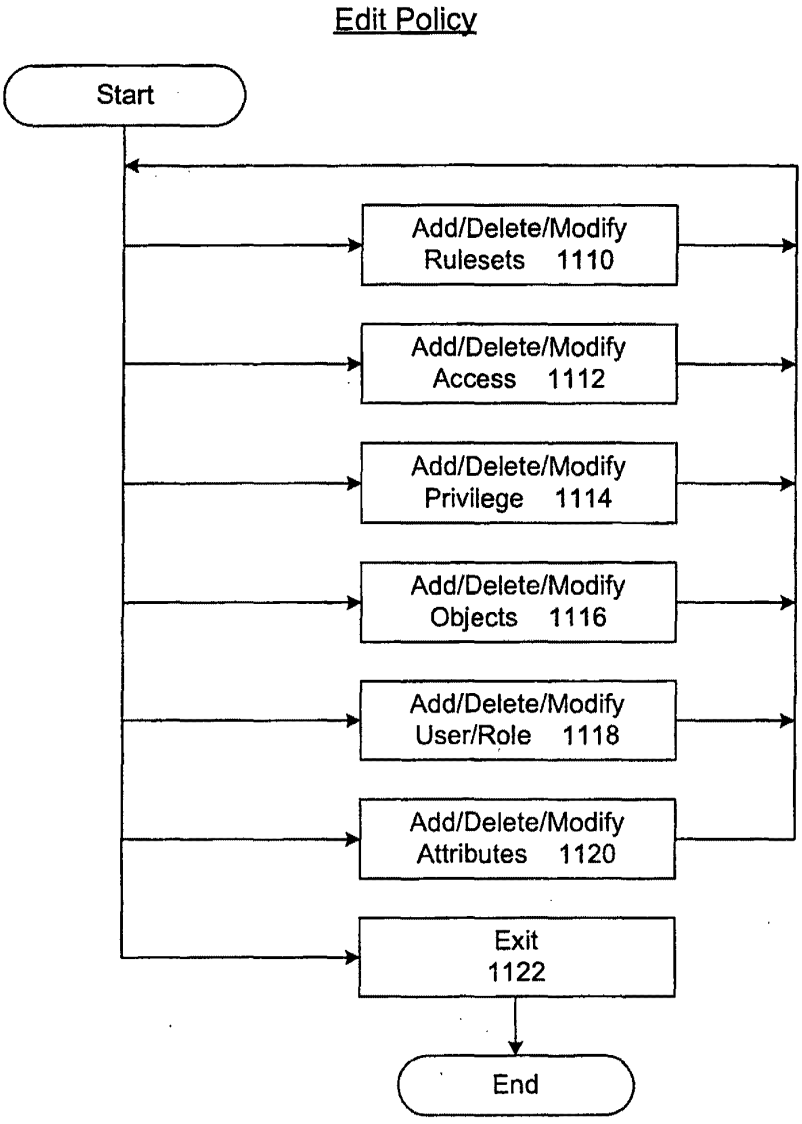

Edit Policy
Start
Add/Delete/Modify Rulesets 1110
Add/Delete/Modify Access 1112
Add/Delete/Modify Privilege 1114
Add/Delete/Modify Objects 1116
Add/Delete/Modify User/Role 1118
Add/Delete/Modify Attributes 1120
Exit 1122
End

FIG. 11

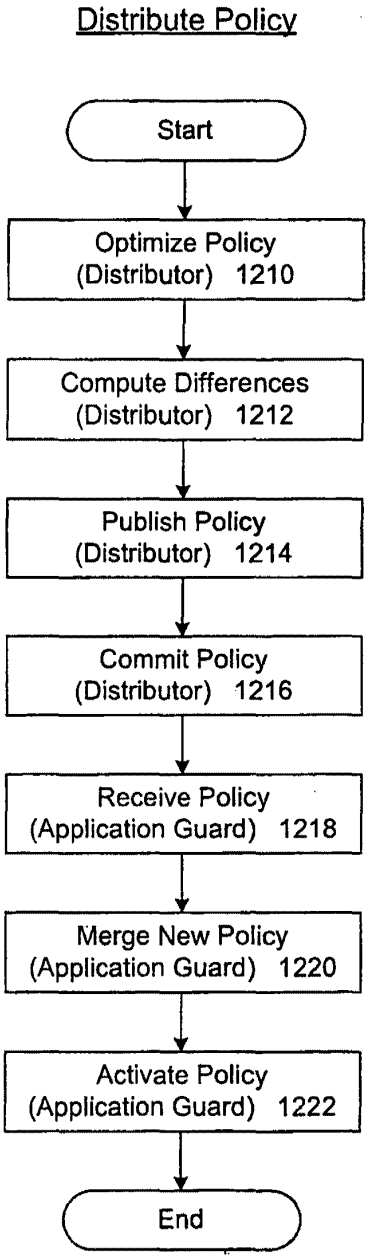

Distribute Policy
Start
Optimize Policy (Distributor) 1210
Compute Differences (Distributor) 1212
Publish Policy (Distributor) 1214
Commit Policy (Distributor) 1216
Receive Policy (Application Guard) 1218
Merge New Policy (Application Guard) 1220
Activate Policy (Application Guard) 1222
End

FIG. 12

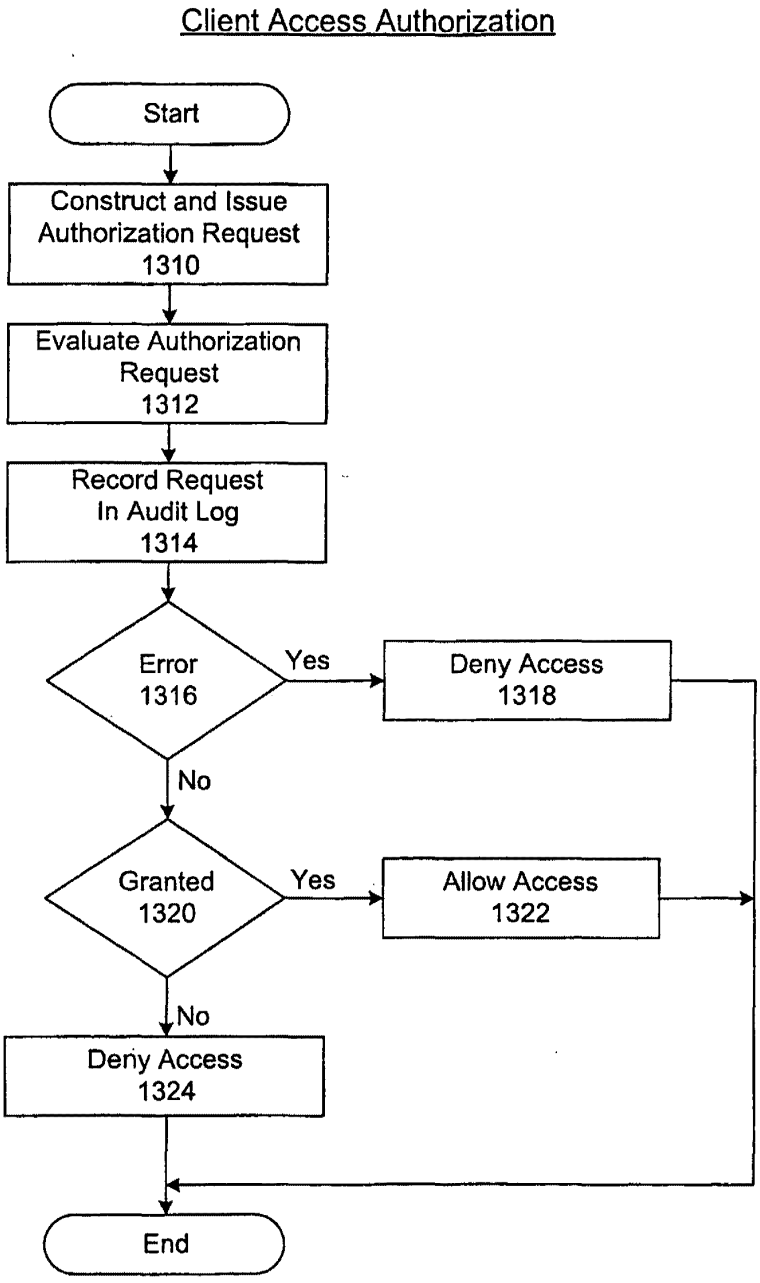

Client Access Authorization
Start
Construct and Issue Authorization Request 1310
Evaluate Authorization Request 1312
Record Request In Audit Log 1314
Error 1316
Yes
Deny Access 1318
No
Granted 1320
Yes
Allow Access 1322
No
Deny Access 1324
End

FIG. 13

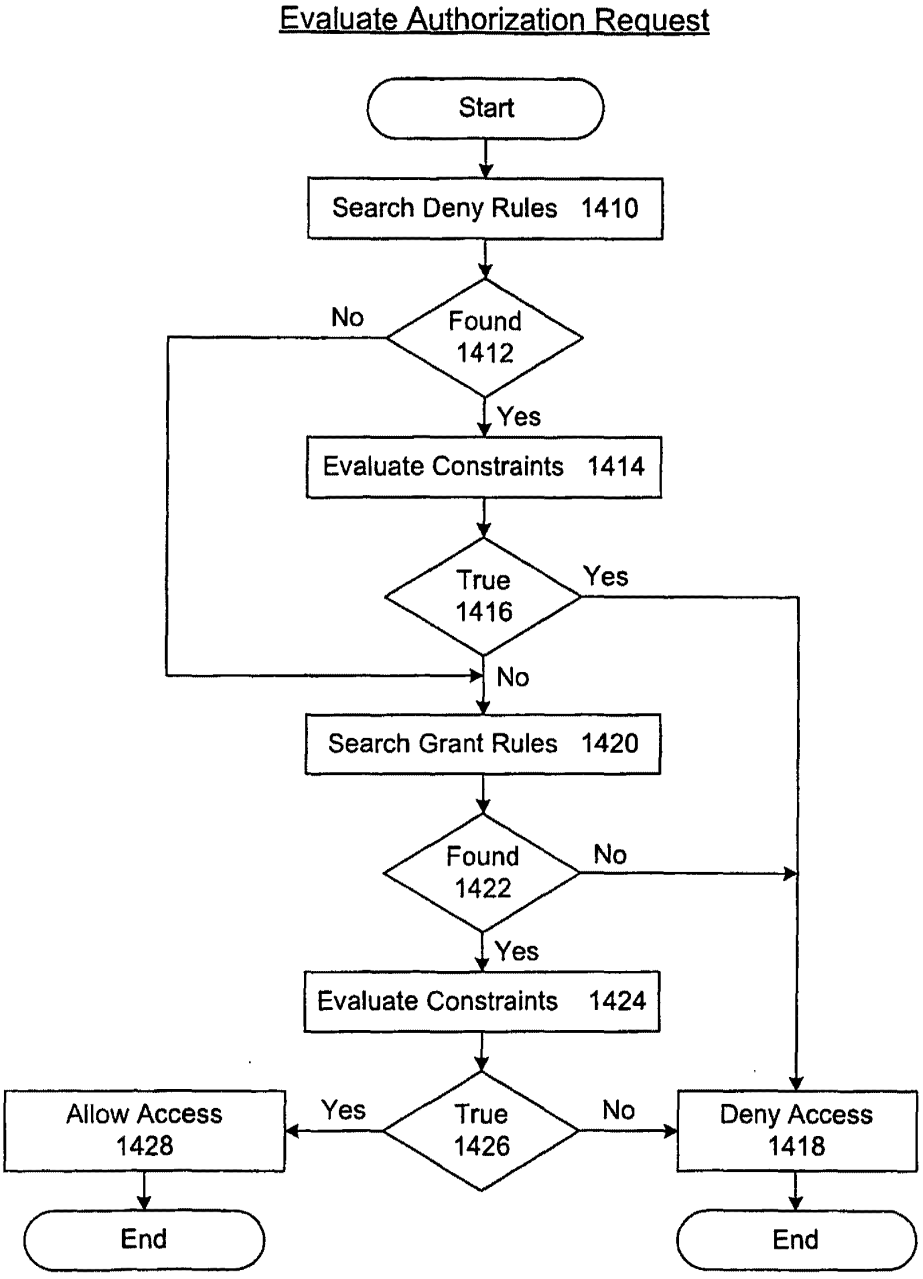

Evaluate Authorization Request
Start
Search Deny Rules 1410
Found 1412
No
Yes
Evaluate Constraints 1414
True 1416
Yes
No
Search Grant Rules 1420
Found 1422
No
Yes
Evaluate Constraints 1424
Allow Access 1428
Yes
True 1426
No
Deny Access 1418
End
End

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/25455

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G 06 F 12/14
US CL : 713/201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 713/201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

713/200, 713/201, 713/202, 709/223, 709/224, 710/39, 710/40

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,E	US 5,968,176 A (NESSETT et al) 19 October 1999, col 8 lines 7-37, col 6 lines 12-34, col 9 lines 7-27, col 12 lines 45-56, col 17 lines 26-31, col 7 lines 22-61, col 5 lines 65- col 6 line 3, col 3 lines 58- 63, col 6 lines 32-34	1-4,6,7, 9-12,17-21 22,24-26, 28,30,31 34-40, 42-49, 51-56, 5,14,15,22 32,41,45 49,13,27

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
16 DECEMBER 1999

Date of mailing of the international search report
03 FEB 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
Joseph Palys James R. Matthews
Telephone No. (703) 305-9605

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/25455

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,E	US 5,983,350 A (MINEAR et al) 09 November 1999, col 5 line 65- col 6 line 3, col 6 lines 15-20, col 11 lines 46-57.	10,19,26, 36, 17,30 47,18,42, 31,53,54 56,48,55
Y,E	US 5,983,270 A (ABRAHAM et al) 09 November 1999, col 7 lines 38-43, col 9 lines 1-10, COL 13 LINES 32-38, COL 12 LINES 15-41.	5,14,22,32 15,41,49 45
Y	US 5,557,747 A (ROGERS et al) 17 September 1996, col 6 lines 47-50	13,27

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

IPR2024-00027
CrowdStrike Exhibit 1002 Page 644

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04Q 3/00 — A2

(11) International Publication Number: WO 00/25527

(43) International Publication Date: 4 May 2000 (04.05.00)

(21) International Application Number: PCT/SE99/01696

(22) International Filing Date: 24 September 1999 (24.09.99)

(30) Priority Data:
09/181,583 28 October 1998 (28.10.98) US

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S–126 25 Stockholm (SE).

(72) Inventors: TSE, Edwin; 4976 Jean Brillant, Montreal, Quebec H3W 1T7 (CA). GOSSELIN, Nicolas; 110 du Blainvillier, Blainville, Quebec J7C 4Y1 (CA).

(74) Agent: ERICSSON RADIO SYSTEMS AB; Common Patent Dept., S–164 80 Stockholm (SE).

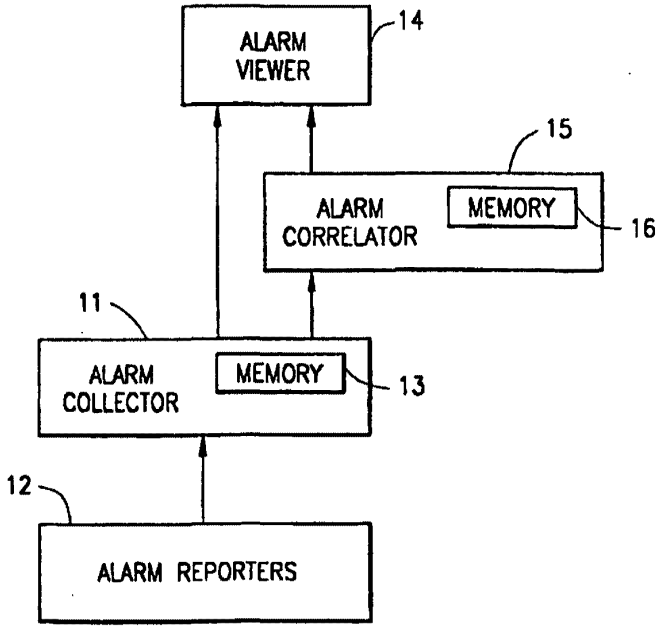
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: ALARM CORRELATION IN A LARGE COMMUNICATIONS NETWORK

(57) Abstract

A system and method of correlating alarms from a plurality of network elements (NEs) in a large communications network. A plurality of uncorrelated alarms are collected by an alarm collector (11) from alarm reporters (12). An alarm correlator (15) then partitions the alarms into correlated alarm clusters (61–63) such that alarms of one cluster have a high probability that they are caused by one network fault. The partitioning of the alarms is performed by creating alarm sets, expanding the alarm sets into alarm domains, and merging the alarm domains into alarm clusters if predefined conditions are met. The sets are formed by selecting an alarmed NE at the highest network hierarchy level which is not tagged, finding all of its contained NEs, and finding NEs that are peer–related to those contained NEs that are in an alarmed state (31–39). The sets are expanded into domains by finding NEs that are not in an alarmed state which contain the highest level alarmed NE in each alarm set (41–47). The domains are merged into one alarm cluster if the two domains have at least one common NE, at least one of the common NEs is not tagged, and the majority of the NEs contained by the non–tagged common NE are in an alarmed state (51–59).



IPR2024-00027
CrowdStrike Exhibit 1002 Page 645

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

ALARM CORRELATION IN A LARGE COMMUNICATIONS NETWORK

BACKGROUND OF THE INVENTION

Technical Field of the Invention

This invention relates to software fault management systems and, more particularly, to a method of correlating multiple network alarms in a large communications network.

Description of Related Art

In communications networks, a single network fault may generate a large number of alarms over space and time. In large, complex networks, simultaneous network faults may occur, causing the network operator to be flooded with a high volume of alarms. The high volume of alarms greatly inhibits the ability to identify and locate the responsible network faults.

In the 1997 IEEE paper, *Fault Isolation and Event Correlation for Integrated Fault Management*, the authors, S. Katker and M. Paterok, describe a state-of-the-art algorithm for alarm correlation. The Katker and Paterok algorithm, however, has several disadvantages. First, the algorithm processes alarms very inefficiently. As noted above, a single fault may trigger a large number of network alarms. For example, one fibre cut can result in hundreds of thousands of alarms being reported from circuits supported by the fibre. The Katker and Paterok algorithm initiates a large number of computing threads, each of which ultimately results in the same conclusion. Thus, an excessive amount of time and computational resources are utilized. Additionally, the Katker and Paterok algorithm fails to correlate network element (NE) alarms that are caused by a faulty NE that does not itself generate an alarm.

In order to overcome the disadvantages of existing solutions, it would be advantageous to have a system and method of correlating large numbers of network alarms which greatly reduces the time and computational resources utilized, and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 647

supports near real-time alarm correlation. The present invention provides such a system and method.

SUMMARY OF THE INVENTION

In one aspect, the present invention is a system for correlating alarms from a plurality of network elements (NEs) in a large communications network. The system comprises a plurality of alarm reporters that report alarms from the NEs when faults are detected, and an alarm correlator that partitions the alarms into correlated alarm clusters such that alarms of one cluster have a high probability that they are caused by one network fault.

In another aspect, the present invention is a method of correlating alarms from the NEs in a large communications network. The method includes the steps of collecting a plurality of uncorrelated alarms from the NEs, and partitioning the alarms into correlated alarm clusters such that alarms of one cluster have a high probability that they are caused by one network fault. The step of partitioning the alarms into correlated alarm clusters may include the steps of creating alarm sets, expanding the alarm sets into alarm domains, and merging the alarm domains into alarm clusters if predefined conditions are met. The domains are merged into one alarm cluster if, and only if, the two domains have at least one common NE, at least one of the common NEs is not tagged, and the majority (as defined by the network operator) of the NEs contained by the non-tagged common NE are not in an alarmed state.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood and its numerous objects and advantages will become more apparent to those skilled in the art by reference to the following drawings, in conjunction with the accompanying specification, in which:

FIG. 1 is a time line illustrating the sequence of tasks for network fault management;

FIG. 2 is a functional block diagram illustrating the data flow between functional blocks of the present invention;

FIG. 3 is an illustrative drawing of two simple span relations;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 648

FIG. 4 is an illustrative drawing of a simple contain relation;

FIGS. 5A-5C are a flow chart illustrating the steps of the preferred embodiment of the method of the present invention; and

FIG. 6 is an illustrative drawing of the relationship between alarm domains and alarm clusters.

DETAILED DESCRIPTION OF EMBODIMENTS

The present invention is a method of partitioning alarms into sets in such a way that each set of alarms is probably caused by the same fault. The method utilizes network alarm propagation characteristics, alarm cause codes, network hierarchy (digital/analog hierarchy and equipment containment hierarchy), and network topology information to partition the alarms into sets. Presentation of alarms in partitioned sets reduces the time required to locate network faults so that more time can be spent fixing them. By assigning network operators to locate network faults based on alarm sets rather than alarm severity, alarm type, or reporting location, duplication of effort is reduced. The present invention greatly reduces computational time and resources required to provide rapid, consistent, and repeatable alarms correlation. Thereafter, the network operator may perform known fault location and identification techniques in a more accurate and efficient manner. The method of the present invention is simple, does not depend on experienced network operator knowledge to be captured and tailored to each network, and can be executed in parallel to increase speed.

FIG. 1 is a time line illustrating the sequence of tasks for network fault management. At time point A, a network fault occurs. The fault management process must then perform five tasks 1-5 in order to restore the network to full operational capability. The tasks are to (1) collect alarms and alert the operator of the alarms caused by the network fault, (2) correlate alarms, (3) locate the fault(s) causing the alarms, (4) test to confirm fault location, and (5) repair the faults and restore the network. At time point B, the paying customer perceives that lower quality of service is being provided. The objective of the fault management process, then, is to restore the network before the customer perceives any drop in quality of service. Thus, the more rapidly tasks 1-5 can be performed, the greater the likelihood that the network

IPR2024-00027
CrowdStrike Exhibit 1002 Page 649

can be restored in time.

The present invention performs three major tasks to achieve its overall objectives. First, the method partitions reported network alarms into related sets (called correlated alarm sets) such that alarms of one set have a high probability that they are caused by one network fault. Second, the method displays the correlated alarms in a list in an Alarm Viewer (AV). Correlated alarms of one set are sorted based on severity, time, etc. and then sequentially listed in the AV. Third, the method displays the alarms in the AV in graphical form. Alarmed managed objects whose alarms are in one correlated alarm set are displayed as nodes of a graph. Non-alarmed managed objects that propagate alarms within the set are displayed as different nodes of the same graph. If there are multiple network faults, then multiple graphs are displayed.

FIG. 2 is a functional block diagram illustrating the data flow between functional blocks of the present invention. An Alarm Collector (AC) 11 collects uncorrelated alarms from various Alarm Reporters (AR) 12 and stores them in its persistent memory 13. The alarm reporters may be part of a NE or external to a NE. Alarm reporters report alarms to the alarm collector using a normalized format. Operators can view these uncorrelated alarms using the Alarm Viewer (AV) 14. The Alarm Correlator (ACOR) 15 reads uncorrelated alarms from the AC, separates the alarms into correlated alarm sets, and stores them in its cache memory 16. The alarm viewer, on demand by the operator, fetches correlated alarm sets for display.

The alarm correlator 15 is not event driven in the sense that the alarm correlator will not start a thread (of execution) based on each incoming alarm. In a large network, an alarm rate of 1 alarm per second is not uncommon. In case of an alarm storm caused by a high capacity facility failure (a situation when the alarm correlation method is most useful), the alarm rate can be much higher. Execution threads, spawned at such a rate, may not be able to do meaningful work since they all are competing for limited system resources. Instead, the alarm correlator takes a "snap shot" of all alarms in the network at any moment, correlates the alarms in multiple passes into alarm clusters, and processes each correlated alarm cluster in one thread.

Before execution of the thread, the alarm correlator 15 has NE information in

IPR2024-00027
CrowdStrike Exhibit 1002 Page 650

its cache memory 16. Therefore, the correlation method can proceed at memory speed. There is no requirement for the alarm correlator, while executing the thread, to query other systems or the network to retrieve required information, such as NE relations.

The correlation method of the present invention utilizes several types of input information. The types of relationships among NEs (managed objects) are important inputs to the methodology. Three key types of NE relationships are the span relation, the contain relation, and the peer relation. Examples of span relations are "Circuit-XYZ spans 6 routing nodes" or "Circuit-ABC spans 8 facilities".

FIG. 3 is an illustrative drawing of two simple span relations. The span relation is uni-directional and one-to-many. An example of a span relation is "Circuit-XYZ spans 6 routing nodes". In FIG. 3, line A which is drawn through facilities 21 and 22 represents the span relation "Circuit-A spans two facilities". Line B which is drawn through facilities 21, 23, and 24 represents the span relation "Circuit-B spans three facilities".

FIG. 4 is an illustrative drawing of a simple contain relation. The contain relation is uni-directional and is one-to-many. Examples of contain relations are "a card contains 10 ports", "a log contains 200,000 records", and "a DS1 contains 24 DS0 circuits". In FIG. 4, lines C and D are drawn passing through facility 25 and illustrate the contain relation "facility-F contains two circuits".

The peer relation is bi-directional and is one-to-one. Examples of peer relations are "port-5 connects to one end of circuit-xyz" and "Circuit-xyz is used by customer-y".

To assist in partitioning the alarms into correlated alarm sets, the alarm correlator 15 constructs and maintains a database of NEs. This database carries minimal NE information such as name, relations, and alarm states of the NEs. The information in this database may be cached for fast performance. The information is updated whenever the configuration of the real network changes.

FIGS. 5A-5C are a flow chart illustrating the steps of the preferred embodiment of the method of the present invention. The present invention utilizes a three-pass methodology to provide rapid, consistent, and repeatable alarms correlation

IPR2024-00027
CrowdStrike Exhibit 1002 Page 651

(task 2 of FIG. 1). The method correlates alarms and partitions the alarms into correlated alarm clusters containing related alarms. The three passes are described below.

First Pass:

Referring to FIG. 5A, the first pass partitions alarms into sets and reduces the search space for the second and third passes. In an initialization step 30, all NEs are set to be "non-tagged" (i.e., not yet processed). At step 31, the method first determines whether or not there exists an alarmed network element (NE) that is non-tagged. If no such NEs exist, the method skips to the second pass. If one or more such NEs exist, the method selects the one which is at the highest network hierarchy level at step 32. Starting with the selected NE, the method finds all lower level NEs at step 33 that are directly or indirectly contained in the selected NE. At step 34, all of the found NEs and the high level selected NE are all placed in a set. For each alarmed NE in the set, the peer relation is used at 35 to find peer-related NEs. All of the found peer-related NEs are added to the set at 36. All NEs in the set are tagged at step 37. This creates one alarm set at 38. At step 39 it is determined whether or not the alarmed, non-tagged NE was the last alarmed NE that is non-tagged. If not, the method then returns to step 31, and repeats the process for each alarmed NE in the network that is not tagged. If the alarmed, non-tagged NE was the last alarmed NE that is non-tagged, the method proceeds to the second pass. If there are "X" such NEs, then the first pass ends by creating X sets of NEs, and proceeds to the second pass in FIG. 5B.

Second Pass:

Referring to FIG. 5B, the second pass searches for higher level NEs that are not in an alarmed state or tagged, but that contain the highest level alarmed NE in each set. To do so, the second pass begins at step 41 by selecting one of the X sets identified in the first pass. At 42, the method selects the alarmed NE that is at the highest network hierarchy level in the set. The span relation is utilized at 43 to find all span-related NEs. For each found NE, the span relation is utilized again at step 44 to find its span-related NEs. All of the found span-related NEs are added to the set at 45. The procedure in steps 42 to 45 is then repeated at step 46 for all X sets of alarms. The second pass ends at 47 with X sets of alarms called Alarm Domains being created.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 652

The method then proceeds to the third pass in FIG. 5C.

Third Pass:

Referring to FIG. 5C, the third pass analyzes the Alarm Domains and merges two domains into one Alarm Cluster, based on the characteristics of one or more common NEs. The two domains are merged even if the common NE is not in an alarmed state. At step 51, the Alarm Domains are grouped into 2-domain pairs. At step 52, the method then compares the contents of each domain with the contents of the other domain with which it has been paired. The domains are merged into one alarm cluster if, and only if, the two domains have at least one common NE, at least one of the common NEs is not tagged, and the majority (as defined by the network operator) of the NEs contained by the non-tagged common NE are not in an alarmed state.

Thus, at step 53, it is determined whether or not the two domains share at least one common NE. If not, the method moves to step 54, and the domains are not merged. If there is at least one common NE, the method moves to step 55 where it is determined whether or not the common NE is tagged. If the common NE is tagged, the method moves to step 54, and the domains are not merged. If the common NE is not tagged, the method moves to step 56 where it is determined whether or not the majority of the NEs contained by the non-tagged common NE are in an alarmed state. If the majority of the NEs are not in an alarmed state, the method moves to step 54, and the domains are not merged. If the majority of the NEs are in an alarmed state, the method moves to step 57 where the two domains are merged into one Alarm Cluster. The procedure in steps 52 to 57 is then repeated at step 58 to merge all possible domains and create "Y" Alarm Clusters at 59.

FIG. 6 is an illustrative drawing of the relationship between alarm domains and alarm clusters. The figure includes three (3) alarm clusters 61-63. There are a total of six (6) alarm domains. Alarm cluster 61 includes three (3) alarm domains 64-66. Alarm cluster 62 includes one (1) alarm domain 67. Alarm cluster 63 includes two (2) alarm domains 68-69. NEs (represented by small circles) exist in the alarm domains, and at least one NE is in the alarmed state (dark circle) within each alarm domain. The relations between NEs are not shown in the figure.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 653

In this manner, the present invention partitions alarms into Alarm Clusters in such a way that alarms in each cluster are probably caused by the same fault. The correlated alarm clusters can then be determined or calculated in one thread. In addition, by merging two Alarm Domains into one cluster when the common NE is not in an alarmed state, a probable faulty NE is identified even though it is not being reported in an alarmed state.

Alarm propagation characteristics dictate that faults occurring in the highest layers of the network hierarchy will cause related faults in the lower layers. For example, a fault at facility 25 of FIG. 4, will cause faults of the two contained circuits C and D. However, there are situations in which related alarms are not available at correlation time. One such situation is when, due to cost, the operator does not have fault detection devices on resources in the lowest layer. For example, a telephone network operator may not monitor all of its DS0 circuits; it may only monitor its DS1s (higher layer) for alarms. Another such situation is when a first operator leases capacity (in a higher layer) from a second operator in order to support the first operator's own circuits (lower layer). The first operator may never receive alarms from the second operator concerning faults in the leased capacity. It should be noted, however, that although the alarm correlator 15 makes use of alarm propagation from higher layers to lower layers, it does not rely on all related alarms (e.g., faults of higher and lower layers) being available at correlation time.

It is thus believed that the operation and construction of the present invention will be apparent from the foregoing description. While the system and method shown and described has been characterized as being preferred, it will be readily apparent that various changes and modifications could be made therein without departing from the scope of the invention as defined in the following claims.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 654

WHAT IS CLAIMED IS:

1. A system for correlating alarms from a plurality of network elements (NEs) in a large communications network, said system comprising:

a plurality of alarm reporters that report alarms from the NEs when faults are detected; and

an alarm correlator that partitions the alarms into correlated alarm clusters such that alarms of one cluster have a high probability that they are caused by one network fault.

2. The system for correlating alarms of claim 1 further comprising an alarm collector that collects and stores alarms from the alarm reporters and provides uncorrelated alarms to the alarm correlator.

3. The system for correlating alarms of claim 1 wherein the alarm correlator includes:

means for creating an alarm set by selecting an alarmed NE at the highest network hierarchy level which is not tagged, finding all lower level NEs that are contained in the selected alarmed NE, finding all peer-related NEs for each alarmed lower level NE, and placing the highest level alarmed NE and all the found contained and peer-related NEs in the set; and

means for creating additional alarm sets by repeating the steps for creating the alarm set for each alarmed NE in the network which is not tagged.

4. The system for correlating alarms of claim 3 wherein the alarm correlator also includes means for finding faulty NEs which are not reporting alarm states while a majority of their contained NEs are reporting alarmed states.

5. The system for correlating alarms of claim 4 wherein the alarm correlator also includes means for merging two sets into one alarm cluster if the two sets have at least one common NE, at least one common NE is not tagged, and a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 655

majority of the NEs contained by the non-tagged common NE are in an alarmed state.

6. In a large communications network having a plurality of network elements (NEs), a method of correlating alarms from the NEs comprising the steps of:

collecting a plurality of uncorrelated alarms from the NEs; and

partitioning the alarms into correlated alarm clusters such that alarms of one cluster have a high probability that they are caused by one network fault.

7. The method of correlating alarms of claim 6 wherein the step of partitioning the alarms into correlated alarm clusters includes the steps of:

creating a plurality of alarm sets;

expanding the alarm sets into a plurality of alarm domains; and

merging the plurality of alarm domains into alarm clusters if predefined conditions are met.

8. The method of correlating alarms of claim 7 wherein the step of creating a plurality of alarm sets includes the steps of:

creating a first alarm set, said step of creating a first alarm set comprising:

selecting an alarmed NE at the highest network hierarchy level which is not tagged;

finding all lower level NEs that are contained in the selected alarmed NE;

finding all peer-related NEs for each contained lower level NE which is in an alarmed state; and

placing the highest level alarmed NE and all the found contained and peer-related NEs in the first set; and

creating additional alarm sets, said step of creating additional alarm sets comprising:

selecting additional alarmed NEs in the network which are not tagged; and

repeating the steps for creating the first alarm set for each selected

IPR2024-00027
CrowdStrike Exhibit 1002 Page 656

additional alarmed NE in the network which is not tagged.

9. The method of correlating alarms of claim 8 wherein the step of expanding the alarm sets into a plurality of alarm domains includes the step of finding NEs that are not in an alarmed state which contain the highest level alarmed NE in each alarm set.

10. The method of correlating alarms of claim 9 wherein the step of finding NEs that are not in an alarmed state which contain the highest level alarmed NE in each alarm set includes the step of expanding the alarm sets to create alarm domains, the step of expanding the alarm sets including the steps of:

(a) selecting one of the alarm sets;

(b) selecting an alarmed NE which is at the highest network hierarchy level in the selected set;

(c) finding all NEs that are span-related to the selected alarmed NE;

(d) finding all NEs that are span-related to each of the found NEs that are span-related to the selected alarmed NE;

(e) adding all the found span-related NEs to the alarm set to create an alarm domain; and

(f) repeating steps (b) through (e) for all of the alarm sets.

11. The method of correlating alarms of claim 10 wherein the step of merging the plurality of alarm domains into alarm clusters if predefined conditions are met includes the steps of:

determining that two alarm domains have at least one common NE;

determining that the common NE is not tagged; and

determining that a majority of the NEs contained by the non-tagged common NE are in an alarmed state.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 657

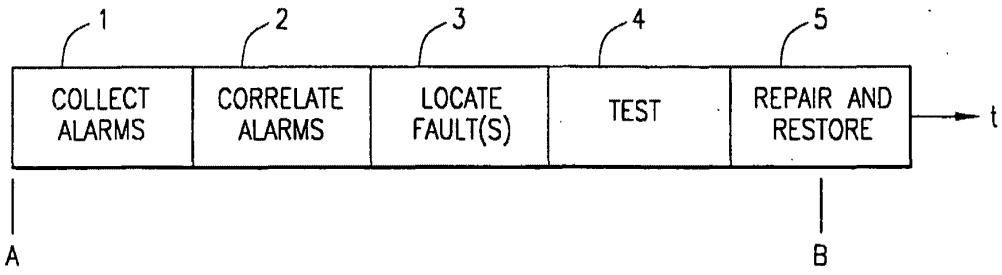

1
2
3
4
5
COLLECT ALARMS
CORRELATE ALARMS
LOCATE FAULT(S)
TEST
REPAIR AND RESTORE
t
A
B

FIG. 1

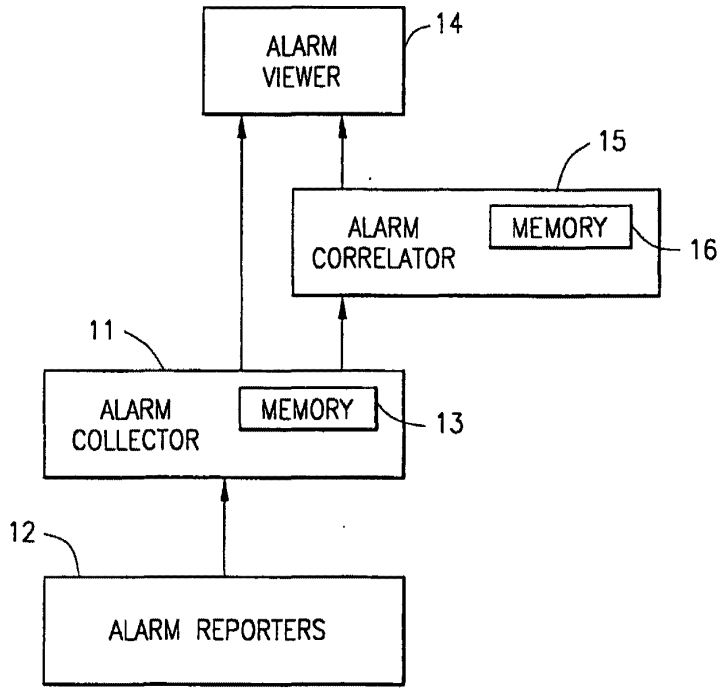

ALARM VIEWER
14
15
ALARM CORRELATOR
MEMORY
16
11
ALARM COLLECTOR
MEMORY
13
12
ALARM REPORTERS

FIG. 2

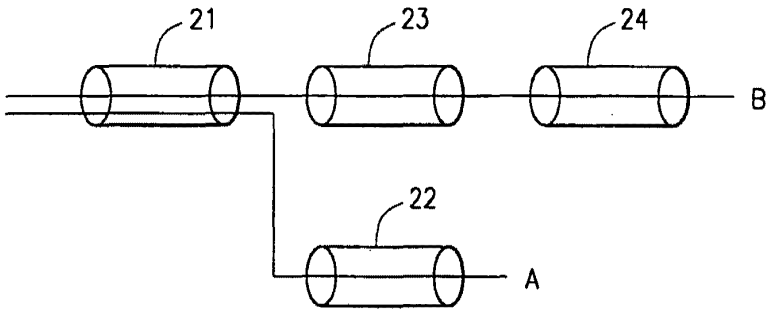


FIG. 3

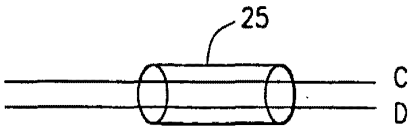
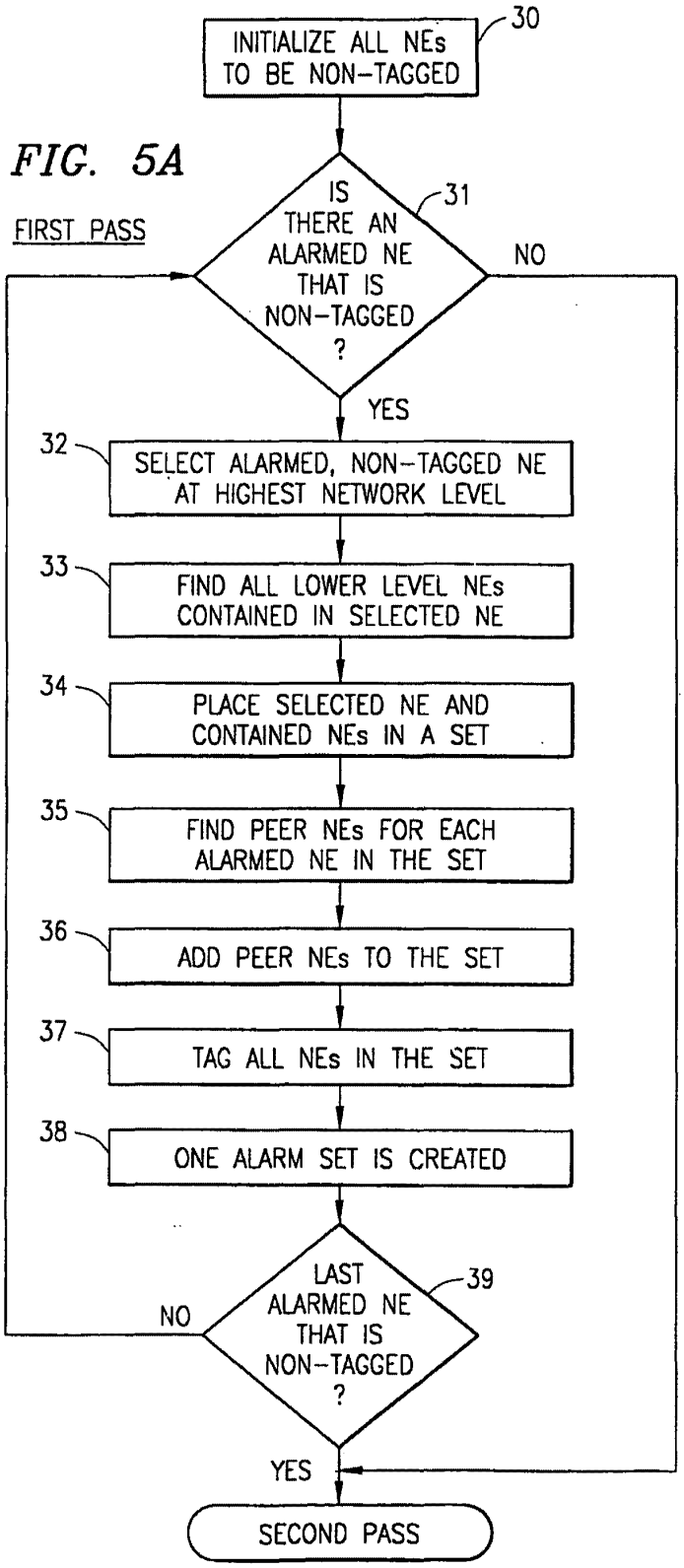


FIG. 4

FIG. 5A



IPR2024-00027
CrowdStrike Exhibit 1002 Page 660

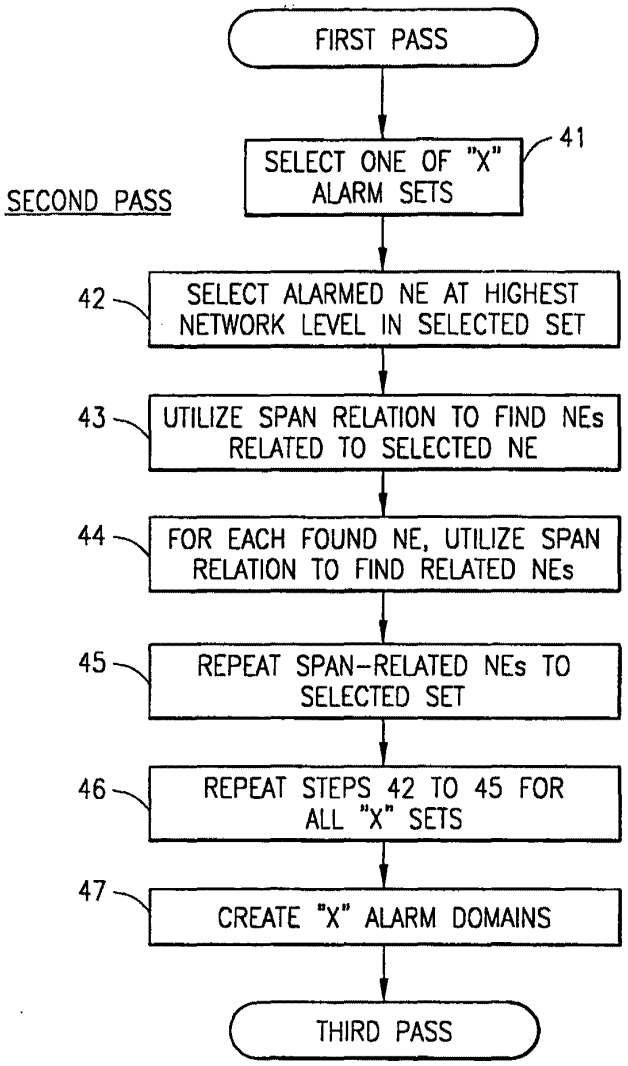
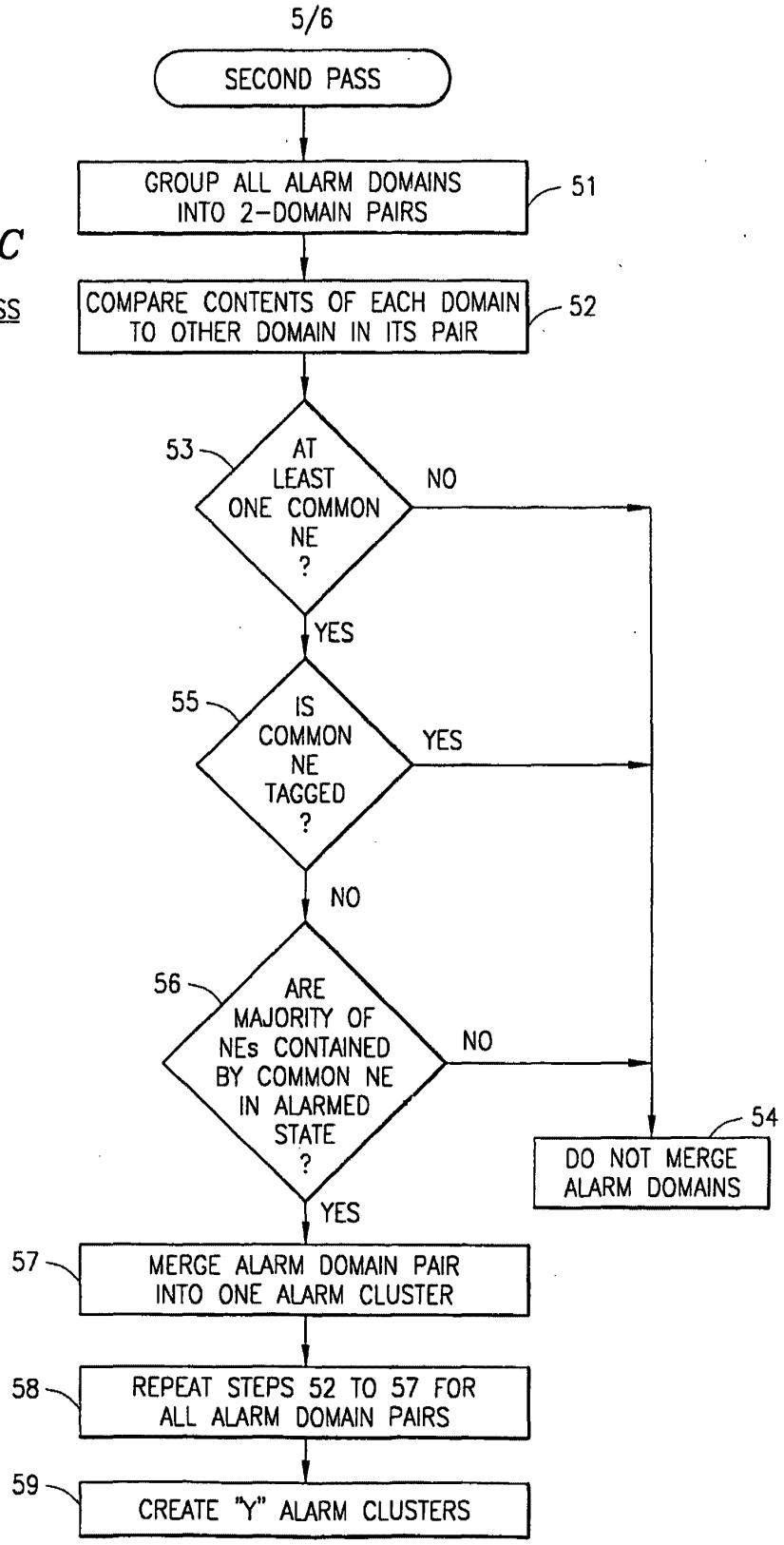


FIG. 5B

IPR2024-00027
CrowdStrike Exhibit 1002 Page 661

FIG. 5C

THIRD PASS



IPR2024-00027
CrowdStrike Exhibit 1002 Page 662

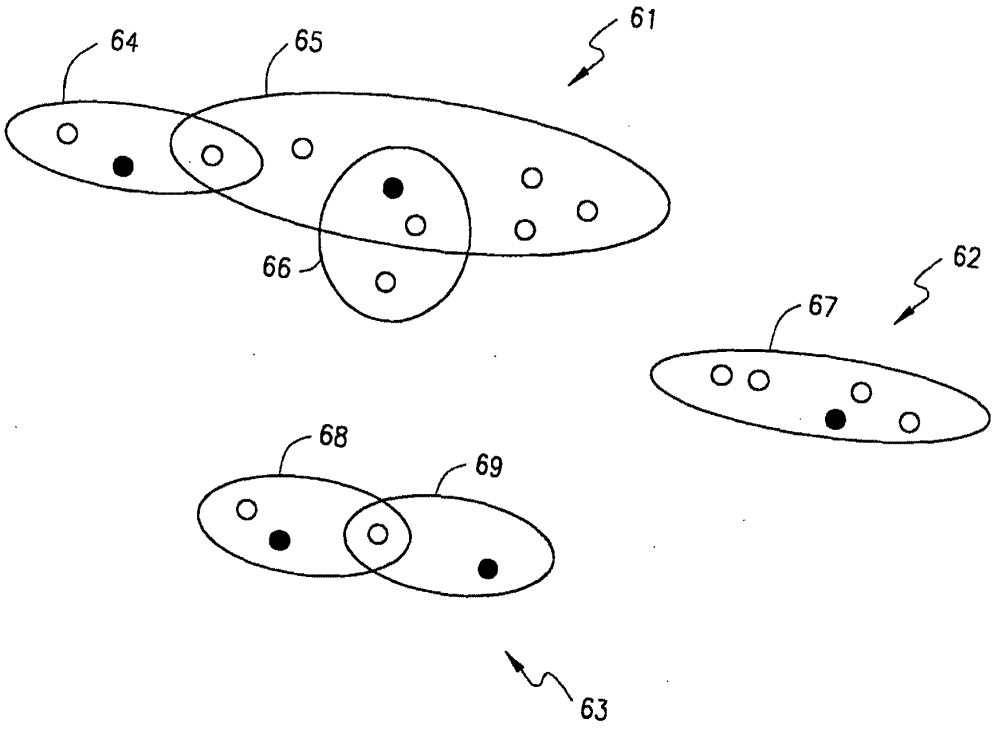


FIG. 6

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 11/30, 12/14, H04L 9/00, 9/32 — A1

(11) International Publication Number: WO 00/34867

(43) International Publication Date: 15 June 2000 (15.06.00)

(21) International Application Number: PCT/US99/28717

(22) International Filing Date: 3 December 1999 (03.12.99)

(30) Priority Data:
60/111,638 9 December 1998 (09.12.98) US
09/447,500 23 November 1999 (23.11.99) US

(71) Applicant *(for all designated States except US)*: NETWORK ICE CORPORATION [US/US]; 30 West 39th Street, Suite 103, San Mateo, CA 94403 (US).

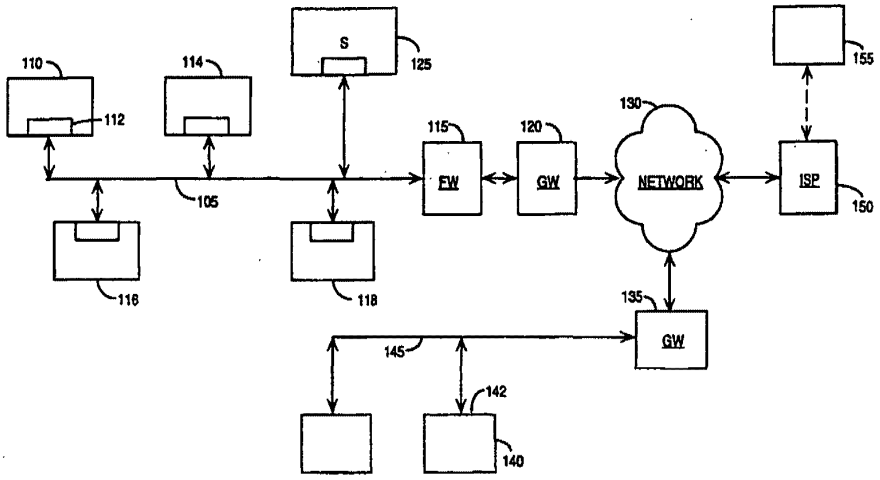
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: GRAHAM, Robert, D. [US/US]; 350 Sharon Park Drive, #H303, Menlo Park, CA 94025 (US).

(74) Agents: MILLIKEN, Darren, J. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A METHOD AND APPARATUS FOR PROVIDING NETWORK AND COMPUTER SYSTEM SECURITY



(57) Abstract

A network intrusion detection and response system (115) and method is disclosed for detecting and preventing misuse of network resources. More particularly, the system and method dynamically self-adjusts to changes in network activity using a plurality of alert levels wherein each successively higher alert level triggers a corresponding heightened security response from the networked computer (125) being misused. These heightened alert levels are integrated on both the system (individual node) and the network level.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 664

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

A METHOD AND APPARATUS FOR PROVIDING NETWORK AND COMPUTER SYSTEM SECURITY

PRIORITY

This application claims the benefit of U.S. Provisional Application No. 60/111,638, filed 12/9/98.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to data network management. More particularly, the invention relates to an improved system and method for detecting and preventing unauthorized use of data network resources.

Description of the Related Art

The rapid increase in the use of data networks by both corporations and private organizations has created a need for improved security and network management techniques. Organizations today store substantial amounts of confidential information on network servers and workstations including trade secrets, marketing strategies, financial documents, and classified technical information. The disclosure of such information to the public would, in most instances, cause severe damage to the organization.

In addition to the danger of confidential information being read out from the network, there is also a danger of unwanted information being *written to* the network. For example, with a working knowledge of how to evade currently available security systems, computer hackers (i.e., unauthorized users) are capable of crashing network servers and workstations, corrupting valuable data, and uploading computer viruses to the network. As such, organizations are forced to spend millions of dollars each year in an attempt to prevent this type of data network intrusion.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 666

One system for handling a type of network misuse is commonly referred to as a "firewall." Firewalls are generally situated between a local area network (hereinafter "LAN") and all other external networks (e.g., the Internet). The firewall analyzes all incoming and outgoing digital information and makes a decision as to whether the information should be passed through or discarded. The firewall uses one or more algorithms provided by a network administrator to perform this analysis. For example, a network administrator may configure tables which list acceptable source and destination addresses for network traffic. Traffic addressed to an unlisted source or destination will be filtered out and discarded by the firewall.

Firewalls provide insufficient protection against computer hackers for a variety of reasons. One major reason is that firewalls only protect LANs from the outside world whereas the threat posed by computer hackers is not merely external. In fact, the majority of potential computer hackers are internal computer users, most of whom already have access to the LAN. Although an individual user will typically be provided only limited access to LAN resources, the user may fraudulently acquire access to additional resources by misappropriating other users' passwords (or using other known computer hacking techniques).

A second problem associated with firewalls is that they are static in nature, requiring continuous updates by network administrators to work properly. If a computer hacker obtains the information necessary to break through the firewall (i.e., information needed to disguise his data as data originating from a legitimate source) he will acquire access to resources on the LAN. Another significant problem with firewalls is that they exclude data in an overly simplistic fashion: data is either passed through or it is discarded. No additional analysis is performed on incoming or outgoing data to determine whether the originator of the data – who may be disguising himself to the firewall – is attempting to misuse resources on the LAN.

One technique used to augment the limited scope of protection provided by firewalls has been referred to as "misuse detection." Misuse detection is the process of monitoring and reporting unauthorized or inappropriate activity on network computers.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 667

For example, Smaha et al., U.S. Patent No. 5,557,742 (hereinafter referred to as "Smaha") discloses a process for detecting a misuse condition by applying predetermined "misuse signatures" to identify known misuses of networked computers. An example of a misuse signature is four unsuccessful logins on a network computer followed by a successful login (see Smaha column 12, lines 12-13).

One limitation of this type of system is that it (like a firewall) is static in nature and requires continuous updates. Thus, as new "misuse signatures" are discovered, they must continually be incorporated into the detection system by programmers or network administrators. Requiring the manual incorporation of new "misuse signatures" is inefficient and will allow an experienced computer hacker access to network resources until his particular "misuse signature" has been determined.

An additional problem with prior art misuse detection systems such as Smaha is that once a potential misuse condition has been observed, the network is limited in its ability to respond to the condition. For example, once a potential hacker has unsuccessfully attempted to login to a networked computer four times as described above, the networked computer will simply deny the hacker access to its resources, rather than taking steps to acquire additional information about the hacker and warn other computers on the network about the hacker. Conversely, if three unsuccessful logins are detected rather than four (followed by a successful login) Smaha discloses no mechanism to raise the system to a heightened security level (i.e., an intermediate level) wherein additional information is collected from the potential hacker before providing access to the network computer.

An additional problem with prior art misuse detection systems such as Smaha is that automated systems can only identify activity as being suspicious, but cannot conclusively differentiate among deliberate misuse attempts, accidents (e.g., user enters the wrong password), or normal incidents (e.g., network manager uses pings to monitor network performance). Thus, prior art misuse detection systems record all suspicious events and rely upon the intelligence of the operator to wade through the "false-positives" in order to find salient records.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 668

SUMMARY OF THE INVENTION

An article of manufacture is disclosed including a sequence of instructions stored on a computer-readable media which when executed by a network node cause the network node to perform the acts of: modifying an alert variable based on data transmissions originating from a suspect node; triggering a first response when said alert variable reaches a first predetermined threshold level; and
triggering a second response when said alert variable reaches a second predetermined threshold level.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

FIG. 1 illustrates generally two local area networks and an Internet Service Provider ("ISP") communicating through a larger network according to one embodiment of the invention.

FIG. 2A illustrates portions of the OSI protocol stack according to one embodiment of the invention.

FIG. 2B illustrates an analysis and filter system implemented between the data link and network layers of the OSI stack according to one embodiment of the invention.

FIG. 3 is a graph illustrating the aggravation response according to one embodiment of the invention.

FIG. 4 is a graph illustrating the overall target aggravation response and the suspect-specific target aggravation response according to one embodiment of the invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 669

FIG. 5 is a graph illustrating the overall target aggravation response and the suspect-specific target aggravation response according to another embodiment of the invention.

FIG. 6 is a graph illustrating the suspect-specific network aggravation response of another embodiment of the invention.

FIG. 7 is a graph illustrating the operation of event overlap according to one embodiment of the invention.

FIG. 8 is a graph illustrating an event data object according to one embodiment of the invention.

FIG. 9 illustrates two time periods which are used to calculate difference reports.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 670

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

What is needed is an improved system and method to aid network administrators in detecting and preventing misuse of network resources. More particularly, what is needed is a comprehensive intrusion detection and response system which will dynamically self-adjust to changes in network activity. What is also needed is an intrusion detection and response system which implements a plurality of alert levels wherein each successively higher alert level triggers a corresponding heightened security response from the networked computer being misused. What is also needed is a system which integrates such heightened alert levels on both the system and the network level. What is also needed is an intrusion detection and response system which can be implemented at low cost and using currently-existing hardware and software (i.e., network computers). What is also needed is an intrusion detection and response system which aggressively filters false-positives while still recording the necessary information to identify an attack.

General Network Configuration

Figure 1 generally depicts a local area network (hereinafter "LAN") 105 over which a plurality of nodes (e.g., 110, 125) communicate. Nodes on LAN 105 include a server 125, a plurality of workstations 110, 114, 116, and 118, and a firewall 115. LAN 105 communicates over a larger network 130 (e.g., the Internet) through a gateway 120. The gateway 120 in one embodiment translates between different network protocols of LAN 105 and network 130 as necessary.

A second LAN 145 and a plurality of nodes 140 are illustrated communicating over network 130 through a second gateway 135. An Internet Service Provider 150 (hereinafter "ISP") is shown communicating across network 130. Node 155 may communicates with ISP 150 in a variety of ways including, for example, a standard telephone connection or an ISDN or cable modem connection.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 671

Each workstation 110, 114, 116, and 118 and server 125 on LAN 105 is a computer comprising a processor and a memory with which software implementing the functionality of the intrusion detection system described herein is executed. Such a computer system stores and communicates (internally or with other computer systems over a network) code and data using machine readable media, such as magnetic disks, random access memory, read only memory, carrier waves, signals, etc. In addition, while one embodiment is described in which the parts of the present invention are implemented in software, alternative embodiments can implement one or more of these parts using any combination of software, firmware and/or hardware.

Each workstation 110, 114, 116, and 118 and server 125 also includes a network interface 112 comprised of hardware and/or software which allows each of the networked computers 110 114, 116, and 118 and server 125 to transmit data over LAN 105. In one embodiment, the hardware portion of network interface 112 is a network interface card connected to the I/O bus of the networked computer 110.

Aggravation Levels

A. Target Aggravation

As used herein, an "incident" is the receipt of a particular type of data transmission by a network node (e.g., a workstation 110, 114, 116, 118, server 125, firewall 115, gateway 120, ISP 150, etc.) hereinafter referred to as the "target node" or "target," originating from another network node, hereinafter referred to as the "suspect node" or "suspect." Certain types of incidents are inherently suspicious from a system security point of view. For example, the transmission of an invalid user ID or password from a suspect node to a target node may indicate that an unauthorized user is attempting to gain access to the target node. Moreover, a *series* of login failures over a relatively short period of time creates a substantially greater suspicion of the suspect node than a single login failure. In other words, a single login failure will generally be the result of an authorized user typing his user ID or password incorrectly from the suspect node. However, with each successive invalid login attempt it becomes substantially less likely that the user of the suspect node is simply making a mistake, and more likely that the user is attempting to gain unauthorized access to the target node.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 672

To address these concerns, in one embodiment of the present system a self-modifying alert level is provided for each target node. The alert level, referred to herein as an "aggravation level," changes dynamically in response to incidents arriving at the target node and to the history of incidents which have previously arrived at the target node. The target node may react to new incidents differently depending on its current aggravation level.

For example, **figure 3** is a graph which shows how the aggravation level of a target node in one embodiment of the present system passes through a plurality of threshold levels 310-350. For the purposes of the following discussion, as time increases in **figure 3** (to the right along the x-axis) it is assumed that the target node receives continual invalid login incidents from one or more suspect nodes. Each threshold is a trigger to take some action. For example, as the aggravation level increases, as shown in curve 360, the first aggravation threshold reached is threshold 310. In one embodiment, at threshold 310 the target node begins a passive scan on all incoming incidents. For example, the target node may begin recording concurrent incoming incidents in an incident log file so that the full extent of the intrusion can be identified. Examples of incidents include web server "hits" (file access), mail commands, port scans, and pings from the suspect(s) to the target.

At the next aggravation threshold, threshold 320, the target node of one embodiment will begin actively scanning the suspect nodes causing the incidents, in an attempt to acquire identification information about the suspect nodes. One example of an active scan is the "finger" command. The finger command returns, among other information, the suspect nodes' network addresses and the identity of the users who are currently logged in to the suspect nodes. At this aggravation threshold, the target may also increase its passive scanning for new incidents. Thus, at aggravation threshold 320, the target may begin to actively acquire information about the suspects and also may increase the logging associated with new incidents.

As the target continues to receive login failures from one or more suspects, its aggravation level reaches threshold 330. Here, the target of one embodiment begins a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 673

more aggressive active scan of the suspects. For example, using the "traceroute" command the target may attempt to ascertain the complete network route taken by data originating form the suspect. Thus, referring once again to **figure 1**, if the target node is node 110 on LAN 105 and the suspect node is node 140 on LAN 145, the traceroute command will trace the communication path between the suspect and the target through network 130 (i.e., it will report the network addresses of all the routers in between). In addition, the target may query the suspect's local gateway – gateway 135 in the present example – for the suspect's data link address (a.k.a. media access control address) using the Simple Network Management Protocol ("SNMP"). More specifically, identification via SNMP can consist of a "get-next" sweep of the table of the ifEntry field (usually, one entry per interface).

As the target's aggravation reaches threshold 340, the target in one embodiment will take additional steps to ensure that an unauthorized suspect is not provided with access to its resources. At this threshold level the target may require additional authentication information from suspects before providing the suspects access. For example, when a suspect transmits *correct* login information (i.e. the correct user ID and password) to the target, the target may initially return an "invalid login" response back to the suspect. Only if the suspect once again transmits the correct login information will the target provide the suspect access. By forcing a double logon in this manner, the target will prevent suspects from acquiring unauthorized access by using automated login scripts, i.e., scripts which run through a series of user ID's and/or passwords in an attempt to determine a valid combination.

At its highest aggravation threshold, threshold 350, the target has received numerous successive login failure incidents, resulting in an unreasonably high probability that an unauthorized suspect is attempting to gain access to its resources. Therefore, at this aggravation level the target may take the final step of blocking incoming incidents (e.g., from every one, from everyone outside its LAN, from a given set of suspects, etc.), thereby denying access to its resources to all of the suspects. The target may also decide to take active scanning measures beyond simple identification, such sending a "ping-of-death" in an attempt to shut down the suspect(s).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 674

Throughout the preceding discussion, specific target node reactions at predetermined aggravation threshold levels have been described. However, alternative embodiments of the invention can use higher, lower and different types of thresholds, threshold reactions, etc., without departing from the underlying principles of the present invention. For example, while the embodiment described above uses the "finger" command at threshold 320 and the "traceroute" command at threshold 330, any alternative/additional active scanning techniques (e.g., "identd" or "ping" commands) could be implemented at these stages without departing from the scope of the present invention. Moreover, different combinations of target node reactions could be employed as well. For example, additional passive scanning techniques could be implemented at each of the threshold levels 310-340 in conjunction with active scanning techniques such as complete packet capture that logs every byte of traffic from the suspect(s).

Moreover, the various threshold levels described above may be calculated using different mathematical equations. For example, the when a new incident arrives at the target node, the aggravation value associated with the incident "A_I" may simply be added to the current aggravation of the target "A" such that a threshold will trigger only if $A_I + A > T$. Alternatively, an equation such as $.5\ A_I + 2A > .75T$ would take half of the incident aggravation level added to twice the current target aggravation level and compare the result with three-quarters of the threshold level.

In addition, as shown by curve 370 of **figure 3**, different types of incidents may provoke different target reactions over time. For example, while the target aggravation level may remain above a maximum threshold value 350 during a continuous series of login failures, it may decrease over time during, for example, for a continuous series of "pings." A "ping," short for "Packet Internet Groper," is a command used to determine whether a node at a particular network address is online. Thus, it is frequently used to test and debug a network by sending out a packet and waiting for a response. In one embodiment of the present system, a series of pings directed at a target will initially increase the target's aggravation level because a suspect may ping a target to verify the target's network address before attempting to evade or disable the target's security system.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 675

Accordingly, as shown in curve 370, the target may initially react with the same aggravation as if it were receiving login failures (i.e., curves 360 and 370 initially overlap). However, over time the aggravation level may decrease, even if the target continues to receive pings. This is because as time passes it becomes less and less likely that the suspect pinging the target is attempting to gain unauthorized access to the target. For example, the suspect may be running network management software and may simply be monitoring the response time of the target node by sending out a ping every few minutes. Alternatively, a network administrator may ping various nodes on the network to troubleshoot portions of the network. As such, when a suspect pings a target over an extended period of time, it is more likely that the suspect is pinging the target for legitimate purposes.

Furthermore, while curves 360 and 370 illustrate an example when the aggravation level is changed at the same rate for different incident types, alternative embodiments vary the rate of change based on the type of incident.

B. Network Aggravation

In addition to maintaining an aggravation level for each individual target node on the network, an aggravation level may also be maintained for the entire network (hereinafter referred to as the "network aggravation level"). For example, the system may be configured such that server 125 receives each target's current aggravation level. Particularly, server 125 may be configured to query each of the target nodes 110, 114, 116, and 118 on LAN 105 at predetermined time intervals to record each target's current aggravation level. Alternatively, or in addition, each target may actively communicate its current aggravation level to server 125 (without first receiving a request from server 125) at predetermined time intervals or when the target's aggravation level transitions from one threshold to another.

The network aggravation level may be calculated by simply averaging the individual target aggravation levels. Alternatively, it may bear some other mathematical relationship with the target aggravation levels. For example, it may be configured to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 676

respond only to significant short term changes in individual target aggravations (which could indicate that a suspect is initiating an attack on a target).

Regardless of how the network aggravation level is calculated, however, it can be *used* in a number of different ways. For example, a first network aggravation threshold may force all target nodes on LAN 105 to require double logins (as described above). A second network aggravation threshold may be set to block network traffic. For example, if the network aggravation level reaches a particular threshold, server 125 may direct gateway 120 to disable all incoming traffic. In this embodiment, server 125 may be configured to differentiate between target aggravations resulting from internal suspects (i.e., residing on LAN 105) and external suspects (i.e., residing over network 130). To accomplish this, server 125 may maintain an internal network aggravation level and an external network aggravation level. The server in this embodiment would disable data traffic through gateway 120 only if the external suspect aggravation level reached a particular threshold.

C. Suspect-Specific and Overall Target Aggravation

In another embodiment, each target node (i.e., workstation 110 and/or server 125) on LAN 105 maintains a unique aggravation level for each suspect node with which it communicates (i.e., from which incidents have originated). This aggravation level will be referred to herein as the "suspect-specific aggravation." In addition to maintaining a suspect-specific aggravation for each individual suspect node, the target node may also maintain an overall aggravation level, referred to herein as the "target aggravation" or the "overall target aggravation."

Although a correlation exists between the various suspect-specific aggravation levels and the overall target aggravation level during certain periods of time, this correlation does not always exist. More specifically, the overall target aggravation will react primarily to new incidents, whereas the suspect-specific aggravation will react to both new and old incidents.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 677

The relationship between the two different aggravations for one embodiment will now be described with reference to **figure 4**. As shown in **figure 4**, if at time t=0 suspect A begins a series of unsuccessful login attempts to the target node, the target's aggravation level towards suspect A (shown as curve 410) will increase with each unsuccessful attempt. The target's suspect-specific aggravation for suspect A will pass through one or more thresholds 450-455 (as described above with reference to **figure 3**) and will level out at some maximum aggravation value 460 representing 100% aggravation. The target's aggravation towards suspect A will remain at this level as long as continuous login failures are received from suspect A. In fact, even if the login failures from suspect A cease, the suspect-specific aggravation in one embodiment will remain at 100% until the target is reset by a network administrator or an automated reset mechanism.

Also illustrated in **figure 4** is curve 420 representing the overall target aggravation. The overall target aggravation, representing the target's aggravation *in general*, reacts to incoming incidents from *all* suspects. Moreover, the overall target aggravation thresholds (e.g., 450, 454, and 455) may cause different reaction types directed towards all suspects (rather than towards a particular suspect as with the suspect-specific thresholds). Thus, as suspect A continually attempts unsuccessfully to log in to the target, the overall target aggravation initially increases along with its suspect-specific aggravation for suspect A. Thus, when the overall target aggravation (curve 420) reaches aggravation threshold 450, the target may conduct a passive scan of *all* incoming incidents (rather than merely scanning incidents originating from a particular suspect). Likewise, when the overall target aggravation reaches threshold 454 the target may require increased authentication from all suspects and at threshold 455 the target may block communication with all suspects.

Thus, at 470 when a second suspect – suspect B – initially pings the target node, the target node may not respond to the ping because the target's *overall* aggravation level is higher than threshold 455 (i.e., the target is blocking communication with all suspects). Note that the suspect-specific aggravation level for suspect B 430 is low at this point because this is the first time that suspect B has pinged the target.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 678

Eventually, however, the overall target aggravation level begins to decrease. As stated above, the overall target aggravation is affected more by incidents which have not been observed continually over a long period of time, whereas the suspect-specific aggravation reacts to both long term and short-term incidents. The logic behind this behavior is that when the target is being aggravated by only a single suspect (or a few suspects) over an extended period of time, there is no need for the target to be suspicious of *all* suspects – only the ones causing its aggravation.

Moreover, because the target's suspect-specific aggravation towards suspect A it beyond threshold 455, the target has taken steps to block communication with suspect A. The target, therefore, no longer needs to maintain a heightened overall target aggravation because the one suspect causing the login failures has been blocked out.

The second time that suspect B pings the target (at 480), the overall target aggravation is low enough so that the target may now respond to the ping. As shown in curves 430 and 440, both of suspect B's pings increase the target's suspect-specific aggravation towards suspect B and the overall target aggravation level 420. The increase in aggravation level may be related to the target aggravation level: incident 470 may be more aggravating than incident 480 because the system is at a more sensitive state.

Figure 5 is a graph illustrating the overall target aggravation response and the suspect-specific target aggravation response according to another embodiment of the invention. The primary difference between **figure 4** and **figure 5** is in the behavior of the target's suspect-specific aggravation towards suspect A (illustrated as curve 510). Suspect A in this example is continually pinging the target rather than continually causing login failures. Therefore, due to the different implications of receiving a series of pings from a suspect and a receiving a series of invalid logins from a suspect (described above in more detail with respect to **figure 2**) the target's suspect-specific aggravation towards suspect A eventually decreases over time along with the overall target aggravation.

D. Suspect-Specific and Overall Network Aggravation

IPR2024-00027
CrowdStrike Exhibit 1002 Page 679

Just as each individual target in one embodiment maintains an overall target aggravation level and one or more suspect-specific target aggravation levels, server 125 in one embodiment maintains an "overall network aggravation" level and a separate "suspect-specific network aggravation" level.

The graph in **figure 6** illustrates how the suspect-specific network aggravation level (curve 600) may react to invalid logins and pings originating from suspect A and directed towards target nodes 110, 114, 116, and 118 on LAN 105. At 605, suspect A pings target 110. As a result, target 110's suspect-specific aggravation towards suspect A increases as shown in curve 610, which initially coincides with the suspect-specific network aggravation level towards suspect A (curve 600). Suspect A's ping causes the suspect-specific network aggravation level to cross threshold 660.

As previously described, threshold 660 may be set to trigger any number of responses from server 125 and/or target nodes 110, 114, 116, and 118. For example, at this threshold server 125 may begin passive scanning of all suspects over LAN 105 or network 130. In another embodiment, server 125 does not conduct any passive scanning of suspect A itself. Rather, when the suspect-specific network aggravation level for a particular suspect reaches a particular threshold which calls for passive scanning, server 125 notifies all target nodes on LAN 105 that they should individually increase their suspect-specific target aggravation towards suspect A.

At 615, suspect A pings target 114 and target 114's suspect-specific aggravation for suspect A increases as shown in curve 620. In response to this second ping by suspect A the suspect-specific network aggravation towards suspect A increases along curve 600 past a second threshold 661. Similarly, at 625, when suspect A unsuccessfully attempts to login to target 116, target 116's suspect-specific target aggravation increases (curve 630), and this incident contributes to the suspect-specific network aggravation towards suspect A. As suspect A continues to ping and/or produce login failures at 635 and 645, the suspect-specific network aggravation also increases as shown in curve 600.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 680

Throughout this process, several additional thresholds 661-664 may be crossed. The same and/or different types of threshold responses described above with reference to the various target aggravation levels can also be implemented for the entire network. Thus, at threshold 661, one or more of the nodes on LAN 105 (e.g., server 125) may begin an active scan of suspect A to acquire additional information about suspect A on behalf of the entire network. At threshold 662, server 125 may begin a more extensive active scan and may also increase its passive scanning of suspect A. At threshold 663, server 125 may require suspect A to provide increased authentication to access any target node on LAN 105 (e.g., double logons). Finally, at the maximum threshold 664, server 125 may block all communication with suspect A over LAN 105. Server 125 may accomplish this either by communicating directly with gateway 120 and firewall 115, or by signaling all targets in LAN 105 to individually block communication with suspect A.

One important feature of the suspect-specific network aggravation response disclosed in **figure 6** is that it increases based on *all* network-wide incidents originating from suspect A. Thus, while the suspect-specific target aggravations towards suspect A (curves 610, 620, 630, 640, and 650) individually remain low, the suspect-specific *network* aggravation response to each of these incidents has a cumulative effect and suspect A is properly identified as a suspicious node.

The suspect-specific network aggravation can be calculated in a variety of ways. In one embodiment, it is merely the sum of all suspect-specific target aggravation levels for suspect A. In another embodiment, it is the *average* of all suspect A suspect-specific target aggravation levels. In still another embodiment, the suspect-specific network aggravation is calculated independently of the individual suspect-specific target aggravation levels. This embodiment might be particularly useful on a LAN with a substantial number of target nodes.

For example, if 250 target nodes reside on LAN 105, suspect A may only make one attempt to gain unauthorized access to the first target and may then move on, making only one attempt at each of the remaining 249 targets. By the time suspect A is ready to make another attempt at the first target, the suspect-specific target aggravation level of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 681

the first target for suspect A might be too low to trigger a threshold due to the time lapse between login failures. However, if a central repository (e.g., server 125) on LAN 105 is following suspect A's activity across the entire network, then the suspect-specific network aggravation level for suspect A should be very high. In fact, if suspect A has attempted unsuccessfully to log in to all 250 targets, then the suspect-specific network aggravation level for suspect A should have surpassed its maximum threshold value, and all network communication with suspect A should be blocked as described above.

Just as each individual target may maintain an "overall target aggravation" which represents the target's aggravation in general, server 125 (or other node on LAN 105) may maintain an "overall network aggravation," which represents the network aggravation in general. Similarly, just as the overall target aggravation responds primarily to new incidents, the "overall network aggravation" in the present embodiment responds primarily to new network-wide incidents.

Thus, as suspect A in the preceding example moves from one target to the next across LAN 105, attempting to log in to each of the 250 individual target nodes, the overall network aggravation level will initially increase along with the suspect-specific network aggravation towards suspect A. However, as time passes and suspect A continues to cause login failures across LAN 105, the overall network aggravation may drop off. The logic here is similar to that for the overall target aggravation: when the network is aggravated by only a single suspect (or a few suspects) over an extended period of time, there is no need for the network to be suspicious of *all* suspects – only those causing the aggravation. Moreover, because the suspect-specific network aggravation towards suspect A in the preceding example will be beyond a maximum threshold, the network has taken steps to block communication with suspect A.

The overall network aggravation may be calculated by taking the average of all the overall target aggravation levels. Thus, in one embodiment, the node maintaining the overall network aggravation level (e.g., server 125) may query each of the targets on the network at predetermined time intervals. Alternatively, each target may automatically communicate its aggravation level to server 125 when its overall aggravation level passes

IPR2024-00027
CrowdStrike Exhibit 1002 Page 682

through a threshold value. In another embodiment, the overall network aggravation is calculated independently of the overall target aggravations.

Throughout the preceding discussion, a specific implementation of network-level reactions at predetermined aggravation thresholds have been described. Depending on the particular configuration, however, different implementations could be employed without departing from the underlying principles of the present invention. For example, while the embodiment described above uses server 125 as the central network repository for calculating and storing network aggravation levels, any node on LAN 105 could provide the same functionality without departing from the scope of the present invention.

Moreover, different mathematical equations may be used to calculate the various threshold triggers. In one embodiment, threshold levels are calculated by combining multiple aggravation variables. For example the equation $A_1 + .75A_{target} + .25A_{network} > T$ can be used to calculate the effective aggravation level using three parts of the overall target aggravation level, one part of the overall network aggravation level and adding this value to the aggravation level associated with the new incident arriving at the target.

Additionally, each incoming incident may increase each of the different aggravation levels described above based on an unlimited number of equations. For example, in one embodiment the equation $A = S*Count_{log}2$ will adjust the aggravation level "A" based on the severity of the incident type "S" and the current incident count "Count."

Throughout the preceding discussion, specific embodiments of the present system have been described as implemented on a network server (e.g., 125) and/or one or more network workstations (e.g., 110). Depending on the particular configuration, however, different implementations could be employed without departing from the underlying principles of the present invention. For example, the server and detection capability could be combined within a hub, switch, firewall (e.g., 115), gateway (e.g., 120), or a promiscuous mode capture device (e.g., node 118 with adapter in promiscuous mode). In this embodiment, the detection system can simultaneously track multiple targets

IPR2024-00027
CrowdStrike Exhibit 1002 Page 683

according to the traffic it observes passing through the device, rather than just the one target it is implemented on. In this implementation, the network aggravation level is calculated from the targets monitored by the device.

Packet-Level Analysis and Filtration

Data transmission across LAN 105 by workstations 110 and server 125 for one embodiment of the present system will now be described with reference to **figure 2A**, which illustrates the different network protocol layers through which data is passed. When a workstation 110 or server 125 on LAN 105 transmits data to another workstation 114 or server 125 on LAN 105 or across network 130 to a node 140 on LAN 145, the data will pass through each of the transmission layers illustrated in **figure 2A**.

The first transmission layer illustrated in **figure 2A** is the physical layer 210. The physical layer 210 represents the actual medium through which the raw digital network traffic flows. For example, the workstations 110 and servers 125 on LAN 105 may be physically connected using numerous different types of media, including coaxial cable, twisted pair cable (e.g., "10 Base-T"), and fiber optic cable. Alternatively, the workstations and servers may be connected via a wireless transmission system (i.e., an RF or infrared transmission system). The layer directly above the physical layer 210 is referred to as the data link layer 220 (a.k.a. the "media access" layer). The data link layer 220 provides the protocol responsible for providing error-free transmission across the physical layer. It accomplishes this by incorporating the data to be transmitted (i.e. data received from the network layer 225) into data frames and then transmitting the frames sequentially across the physical layer 210. "Ethernet" and "Token Ring" are two well known examples data link protocols.

The network layer 225 resides directly above the data link layer 220 and provides network addressing (among other things) for the data to be transmitted. Data is incorporated into network "packets" at the network layer 225, with each packet

IPR2024-00027
CrowdStrike Exhibit 1002 Page 684

containing a source and a destination address in its header. The "IP" portion of TCP/IP, also known as the Internet protocol, is one well known network-layer protocol.

Thus, when a workstation 110 or server 125 receives data which has been transmitted over LAN 105, it receives the data in data link layer frames, each frame containing one or more network layer packets. It then removes the network packets from the data link frames and transmits the packets up through the remaining protocol layers to the application layer 232 (where the application program resides which requested the data or from which data is being requested by another node).

Referring now to **figure 2B**, an analysis module 250 and a filter module 260 which comprise a portion of the intrusion detection and response system are illustrated. The analysis module 250 provides the aggravation level functionality of the present invention. It receives incoming network packets and determines the origin of the transmitted data and the type of data contained in the packet (e.g., ping, login request . . . etc). It then makes a decision as to how to deal with the incoming data based on the current aggravation levels of the target and the network.

For example, referring once again to **figure 4**, if the suspect-specific target aggravation level towards suspect A is above threshold 450 (after the analysis module 250 receives data packets comprising the latest incident from suspect A), then the analysis module 250 may initiate a passive scan of suspect A. If the new incident received from suspect A causes the analysis module to raise the suspect-specific target aggravation level towards suspect A above thresholds 451 or 452, the analysis module 250 may initiate an active scan of suspect A. If the suspect-specific target aggravation level for suspect A reaches threshold 455, the analysis module may require increased authentication from suspect A before providing suspect A access to the target's resources.

Finally, if the suspect-specific target aggravation towards suspect A is above a maximum threshold level, e.g., threshold 455, then analysis module 250 will apply filter module 260 to selectively filter out all data packets received from suspect A. While the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 685

foregoing discussion is focused on the suspect-specific target aggravation level, it should be noted that the analysis module 250 may also react to new incidents based on any of the other aggravation levels previously discussed (e.g., the overall target aggravation, the overall network aggravation, and the suspect-specific network aggravation).

It should be noted that the analysis 250 and filter 260 modules are inserted in the network protocol stack at the packet level (i.e., between the network and data link protocol layers). Thus, every data packet is analyzed before being passed through to the computer on which analysis module 250 and filter module 260 reside. As such, even "stealth" incidents (those designed to evade logging at the transport 230 and/or application 232 layers) are analyzed.

Throughout the preceding discussion, a specific embodiment of packet-level analysis has been described. Depending on the particular configuration, however, different implementations could be employed without departing from the underlying principles of the present invention. For example, while the embodiment described above analyzes network and transport layer connection incidents from packet information, it may instead choose to read the same information from the transport stack. Similarly, instead of recording login failure packets, the system may read those events from the application layer logging and auditing system. Thus, packet-level information can be reconstructed from other logging, auditing, and monitoring subsystems on the target without departing from the scope of the present invention.

Event-Time Rollup

As used herein an "event" is a particular type of data communication (e.g., a ping) sent from a particular suspect to a particular target. An event data structure 800 is illustrated in **figure 8**. The event structure includes an event type field 810 (e.g., ping), a suspect identification field 820, a target identification field 830, a counter field 840, an interval field 850, and a window size field 860. When a target node or server begins passive scanning because one or more of the aggravation levels described above is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 686

beyond a predetermined threshold level, the target node or server may begin to log (e.g., store to hard disk) a record of all incoming events. In order to conserve memory, numerous events may be combined into a single event structure. This procedure, referred to herein as "event-time rollup" will now be described with respect to **figure 7**.

As shown in timeline 710, a window size 860 of time T initially surrounds each event. When the windows of two successive events overlap as shown in timeline 720, the two events are combined into a single event data object 700. Thus, after the second event arrives at t_2 in figure 720, the event data structure 800 will increase it's count data field from 1 to 2, indicating that there were two successive events which overlapped. This method of combining several events into a single data object is done to preserve memory and also to prevent a particular type of attack by a suspect wherein the suspect attempts to disable the target by filling up the target's hard drive will vast amounts of event data.

In addition, the event window of one embodiment may be dynamically expanded and contracted. For example, if events arrive slowly as shown in timeline 730, the event window can be expanded so that future events separated by the same time interval T_1 will be combined in the same data structure 800. For example, the new window may be expanded to twice time interval T_1. Thus, if the first and second events are separated by a 10-minute time interval and the default window T is 15 minutes, then the window will be expanded to 20 minutes. Moreover, if the event rate slowly decreases, then the window will continue to expand to keep up.

Another variable which may effect the event window size is one or more of the aggravation levels discussed above. Generally speaking, events which are classified as more serious (i.e., events that are more likely to be attacks by a suspect based on one or more aggravation levels) will cause the event window to expand. Thus, generally speaking, the default window size for a login failure event will be larger than the default window for a ping event.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 687

In one embodiment, the only window which is considered when determining whether to include successive events in a single data object is the window associated with the *current* incoming event (e.g., the event at t2 in 720 and at t3 in 730). In this embodiment, the only question is whether the previous event (e.g., occurring at t1 at 720 and 730) falls within the window of the new event. Accordingly, in this embodiment, the current window for the new event (i.e., at t2) will generally need to be larger to produce the same effect because the window surrounding t1 will no longer be a factor.

Difference Reports

In addition, periodic reports may be generated which summarize all *new* events which occur during a predetermined time period. For example, **figure 9** illustrates two periods of time, 910 and 920, during which a series of events, 930-960 occur. Period 910 represents a time period for which a report has already been generated. As indicated in **figure 9**, the first event series A 930 represents a series of two or more events separated by an interval I_A. The events included in event series A 930 are all events of the same type, suspect, and target. These events will not be included in the difference report generated at the end of the second time period 920 because event series A contains events which occur in both time periods.

Similarly, event series B 940, 945 will not be recorded because events of the same type, suspect and target occur in both time periods 910, 920. Thus, in one embodiment, the first step in creating a difference report is to determine whether the series of events in question overlap between the two time periods. If they do overlap, then they are not included in the difference report. If they don't overlap, then event data objects in the previous time period are searched to determine if one of the same type, suspect, and target was recorded. If such an event data object is found, then the events are not included in the difference report. Thus, the difference report produces the same result whether or not two events where combined (using the event rollup method) into a single longer-term event, or remained uncombined as two separate events.

By contrast, event C 950 occurs only during the second time period and will therefore show up on the difference report generated at the end of the second period 920.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 688

Event series D will not be included in the difference report generated after the second time period 920 because event series D does not occur during this time period. If a difference report is generated at all for event series D it will be generated at the end of the first time period 910.

The reason for recording only new events (i.e., of the same type, suspect and victim) is that it alleviates the problem of false positives (i.e., identifying an event as suspicious when it should not be identified as such). In other words, an event which occurs during every time period is not generally considered suspicious.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 689

CLAIMS

What is claimed is:

1. An article of manufacture including a sequence of instructions stored on a computer-readable media which when executed by a network node cause the network node to perform the acts of:

modifying an alert variable based on data transmissions originating from one or more suspect nodes;

triggering a first response when said alert variable reaches a first predetermined threshold level; and

triggering a second response when said alert variable reaches a second predetermined threshold level.

2. The article of manufacture as claimed in claim 1 further including the step of triggering additional responses when said alert variable reaches one or more additional threshold levels.

3. The article of manufacture as claimed in claim 1 wherein one of said triggered responses includes a passive scan of one or more of said suspect nodes.

4. The article of manufacture as claimed in claim 3 wherein said passive scan includes the step of recording data transmissions in a log file.

5. The article of manufacture as claimed in claim 1 wherein one of said triggered responses includes an active scan of one or more of said suspect nodes.

6. The article of manufacture as claimed in claim 5 wherein said active scan includes the step of retrieving information about one or more of said suspect nodes including the network address of said suspect nodes.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 690

7. The article of manufacture as claimed in claim 5 wherein said active scan includes the step of determining the network route taken by data originating from one or more of said suspect nodes.

8. The article of manufacture as claimed in claim 1 wherein one of said triggered responses includes said network node requiring increased authentication from any other node before providing access to its resources.

9. The article of manufacture as claimed in claim 8 wherein said increased authentication includes the step of forcing two or more logins before providing access to its resources.

10. The article of manufacture as claimed in claim 1 wherein one of said triggered responses includes the step of blocking incoming data transmissions.

11. The article of manufacture as claimed in claim 1 wherein said alert variable responds differently over time to particular types of data transmissions.

12. The article of manufacture as claimed in claim 11 wherein said alert variable continuously increases in response to the continuous receipt of a particular type of data transmission until the alert variable reaches a predetermined value.

13. The article of manufacture as claimed in claim 12 wherein said particular type of data transmission originating from said suspect node is an invalid login attempt.

14. The article of manufacture as claimed in claim 11 wherein said alert variable initially increases in response to the continuous receipt of a particular type of data transmission and subsequently decreases in response to the continued receipt of said particular type of data transmission.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 691

15. The article of manufacture as claimed in claim 14 wherein said particular type of data transmission originating from said suspect node is a transmission which retrieves information about said network node (e.g., the "ping" command).

16. The article of manufacture as claimed in claim 1 wherein said data transmissions are analyzed by said network node on a network packet level.

17. The article of manufacture as claimed in claim 16 wherein said data transmissions are filtered by said network node on a network packet level.

18. An article of manufacture including a sequence of instructions stored on a computer-readable media which when executed by a network node cause the network node to perform the acts of:

modifying a first suspect-specific alert variable based on data transmissions originating from a first suspect node; and

modifying a second suspect-specific alert variable based on data transmissions originating from a second suspect node; and

triggering a suspect-specific response when either of said suspect-specific alert variables reach a predetermined threshold level.

19. The article of manufacture as claimed in claim 18 including the act of triggering additional suspect-specific responses when either of said suspect-specific alert variables reaches additional predetermined threshold values.

20. The article of manufacture as claimed in claim 18 including the act of modifying an overall alert variable based on said data transmissions originating from each of said suspect nodes.

21. The article of manufacture as claimed in claim 20 including the act of triggering a response towards each one of said plurality of suspect nodes when said overall alert variable reaches a predetermined threshold value.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 692

22. The article of manufacture as claimed in claim 20 wherein said overall alert variable is more responsive to new types of data transmissions than to data transmissions previously received at said network node.

23. The article of manufacture as claimed in claim 22 including the act of initially increasing said overall alert variable in response to data transmissions originating from a particular suspect node and subsequently decreasing said overall alert variable upon continued receipt of said data transmissions from said particular suspect node.

24. The article of manufacture as claimed in claim 18 including the act of communicating each of said suspect-specific alert variables to a network database residing on a server node.

25. The article of manufacture as claimed in claim 20 including the act of communicating said overall alert variable to a network database residing on a server node.

26. An article of manufacture including a sequence of instructions stored on a computer-readable media which when executed by a network server node cause the network server node to perform the acts of:

storing a plurality of suspect-specific alert variables for a plurality of network nodes;

modifying a network alert variable based on the value of each of said plurality of suspect-specific alert variables; and

triggering a network response when said network alert variable reaches a predetermined threshold level.

27. The article of manufacture as claimed in claim 26 wherein said network response includes the act of notifying each of the plurality of network nodes that they should each increase their suspect-specific alert variable towards a particular suspect node.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 693

28. The article of manufacture as claimed in claim 27 wherein said network response includes the act of said network server node initiating a passive scan of a particular suspect node.

29. The article of manufacture as claimed in claim 27 wherein said network response includes the act of said network server node initiating an active scan of a particular suspect node.

30. The article of manufacture as claimed in claim 29 wherein said network response includes the act of blocking all communication between said suspect node and said plurality of network nodes.

31. An article of manufacture including a sequence of instructions stored on a computer-readable media which when executed by a network server node cause the network server node to perform the acts of:

storing a plurality of overall alert variables for a plurality of network nodes;

modifying a network alert variable based on the value of each of said plurality of overall alert variables; and

triggering a network response when said network alert variable reaches a predetermined threshold level.

32. A method comprising:

receiving a first event from a suspect node;

recording said first event in a first data structure having an event count value;

receiving a second event from said suspect node, said second event being of a same type as said first event; and

recording said second event in said first data structure and incrementing said count value if said second event occurs within a predetermined window of time after said first event.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 694

33. The method as claimed in claim 32 further comprising recording said second event in a second data structure having a count value if said second event occurs outside of said predetermined window of time after said first event.

34. The method as claimed in claim 33 wherein said predetermined window of time is increased responsive to said second event occurring outside of said predetermined window of time.

35. The method as claimed in claim 32 wherein said predetermined window of time is modified based on said first or second event type.

36. The method as claimed in claim 35 wherein said window of time is increased for more serious event types and decreased for less serious event types.

37. The method as claimed in claim 36 wherein said event type is an invalid login.

38. The method as claimed in claim 36 wherein said event type is a ping.

39. The method as claimed in claim 32 further comprising generating a report of all new events which occur over a predetermined time period.

40. The method as claimed in claim 39 wherein an event is identified as a new event by:

determining whether said event is included in a single data structure with one or more pervious events received in a time period preceding said predetermined time period;

searching all data structures generated during said time period preceding said predetermined time period if said event is not included in said single data structure with one or more previous events; and

including said event in said report if said event is not identified in any data structures generated during said time period preceding said predetermined time period.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 695

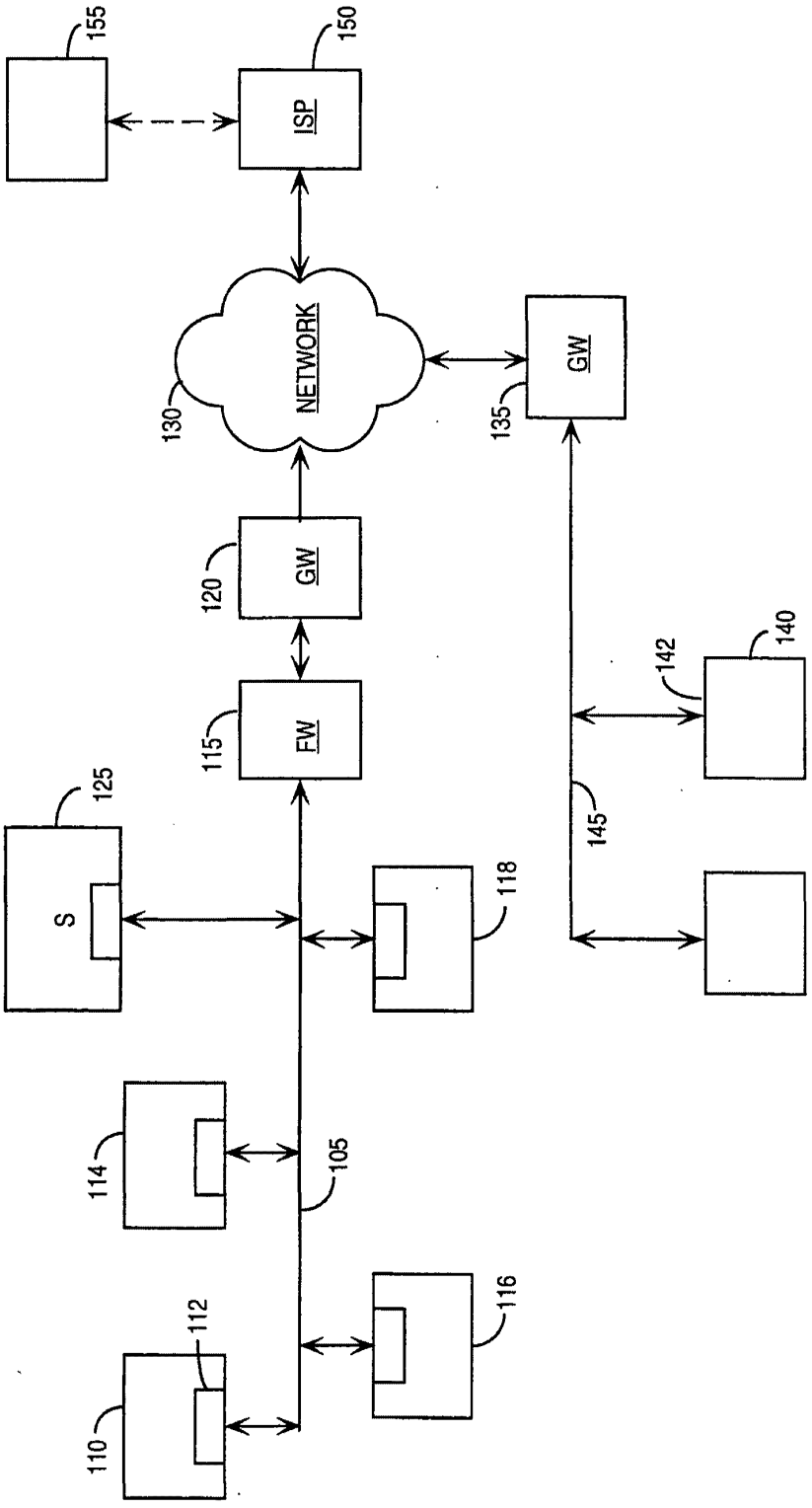


FIG. 1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 696

SUBSTITUTE SHEET (RULE 26)

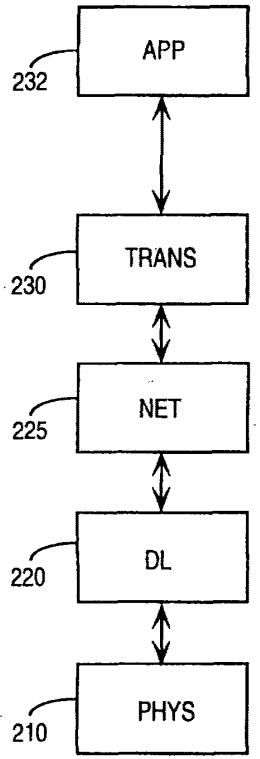


FIG. 2A

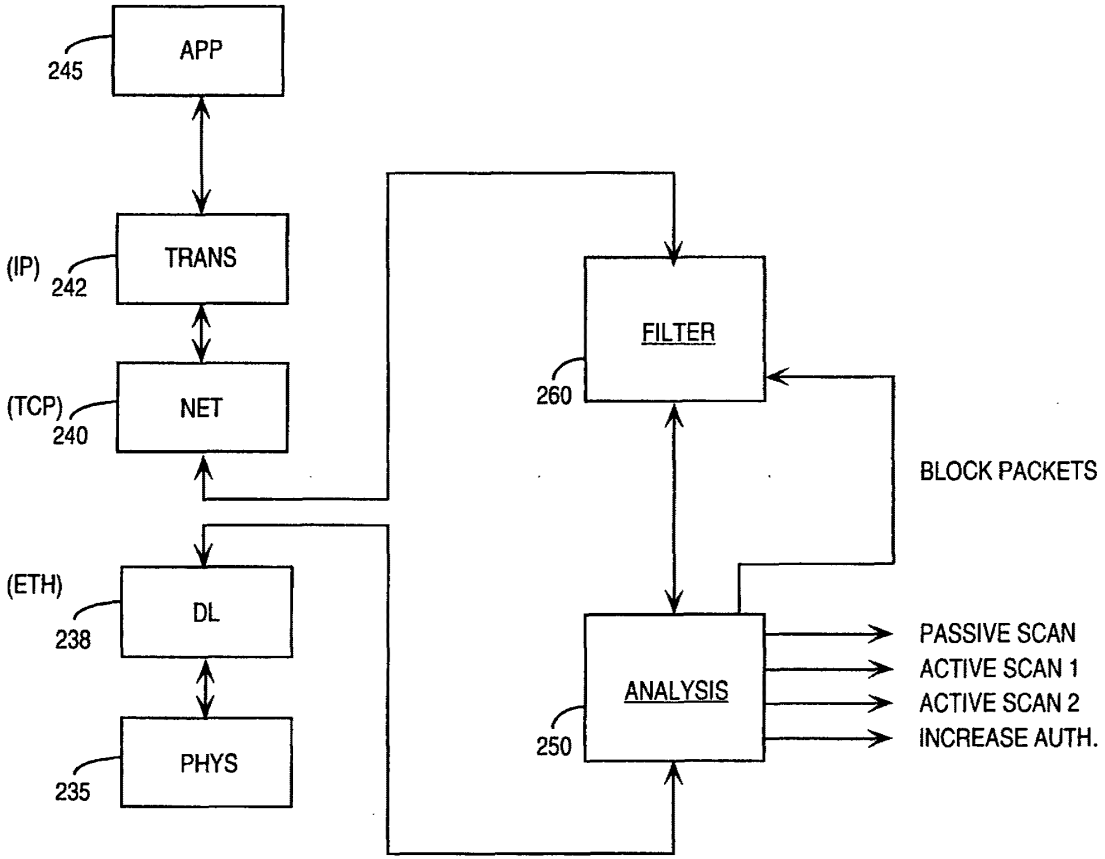


FIG. 2B

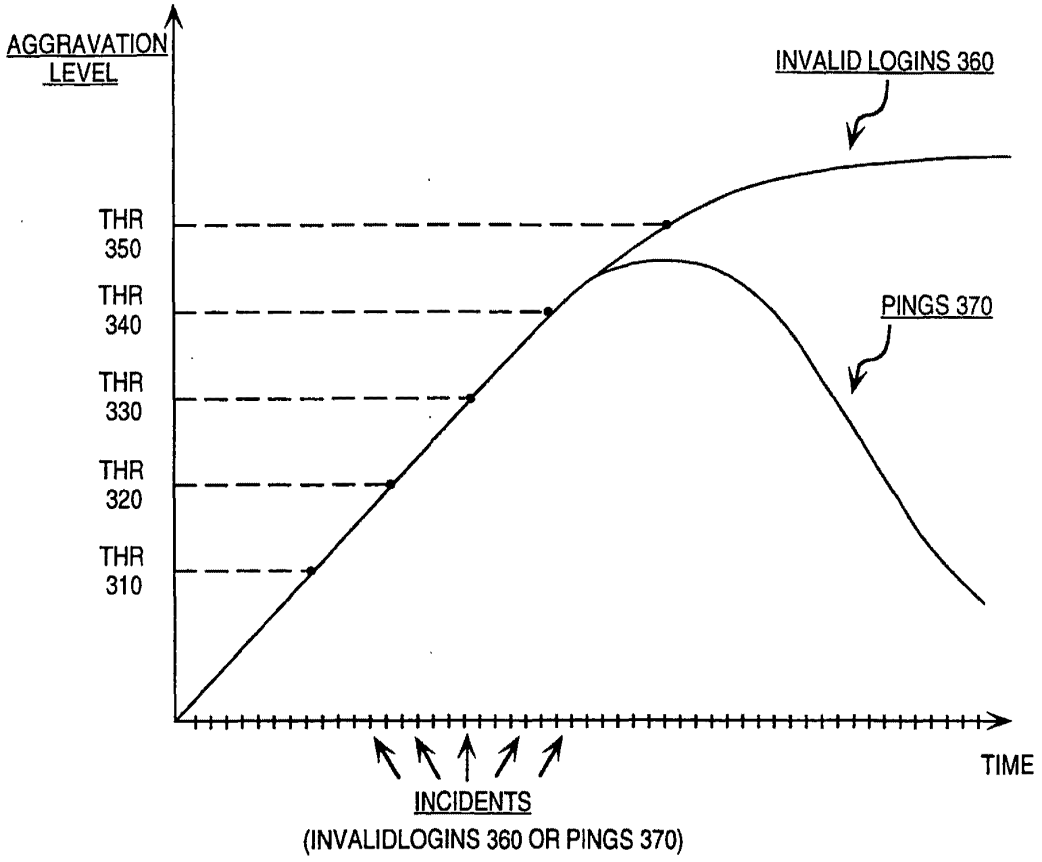


FIG. 3

IPR2024-00027
CrowdStrike Exhibit 1002 Page 698

SUBSTITUTE SHEET (RULE 26)

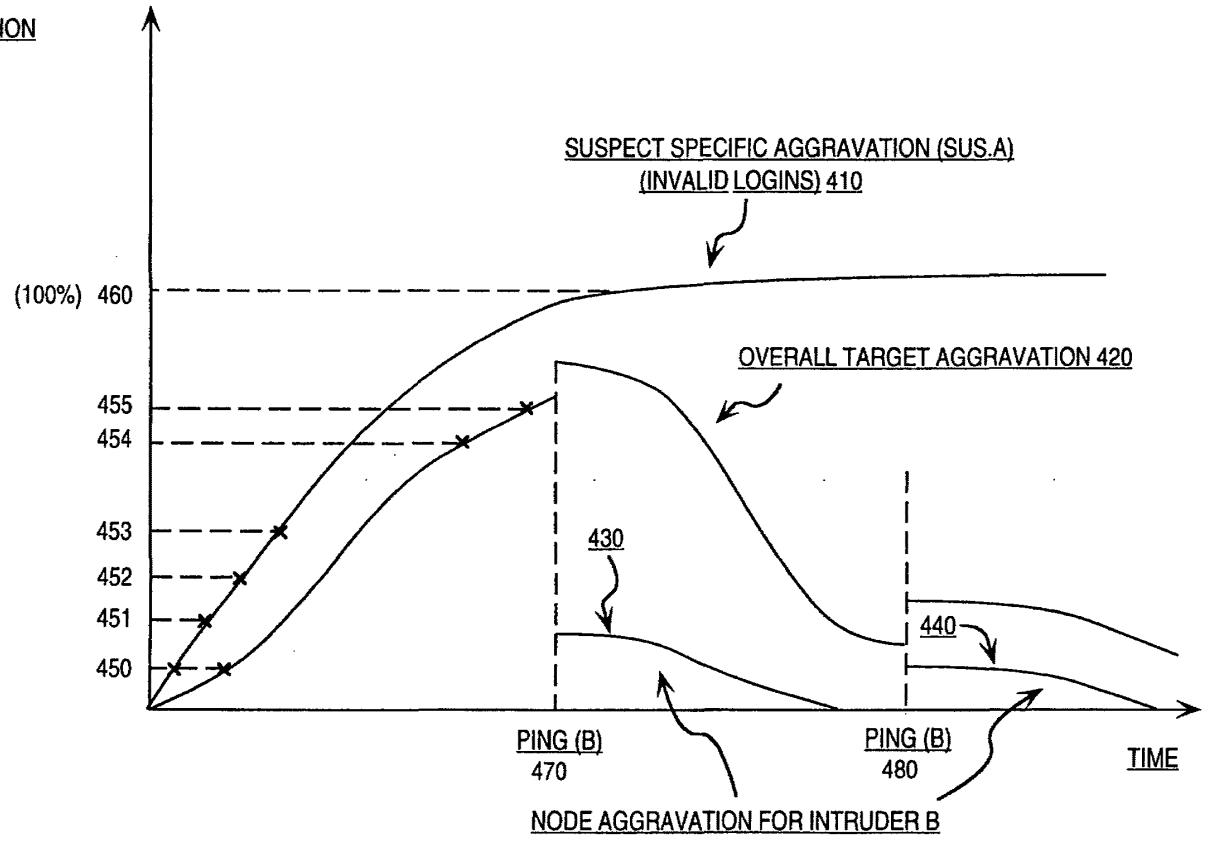


FIG. 4

IPR2024-00027
CrowdStrike Exhibit 1002 Page 699

SUBSTITUTE SHEET (RULE 26)

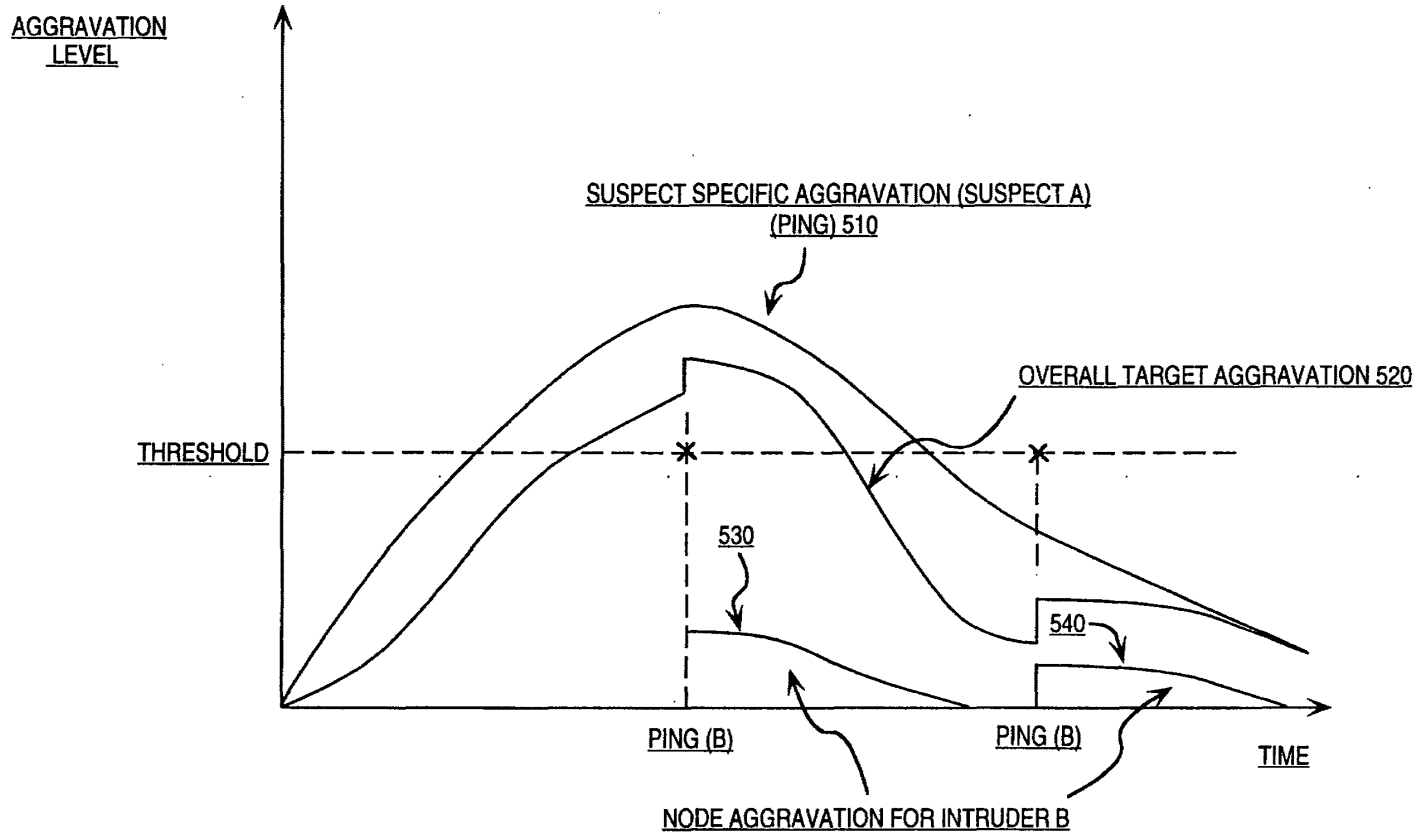


FIG. 5

IPR2024-00027
CrowdStrike Exhibit 1002 Page 700

SUBSTITUTE SHEET (RULE 26)

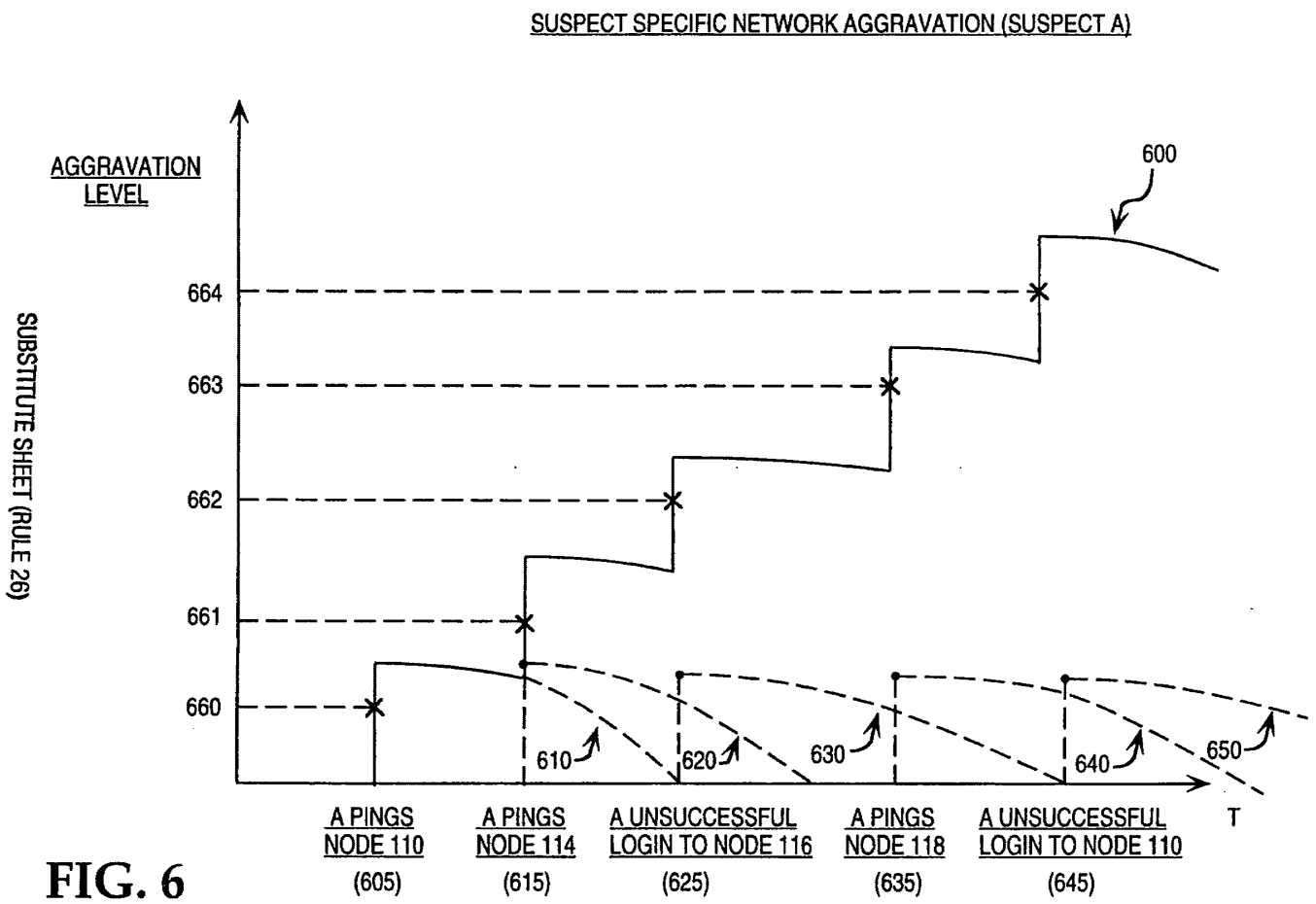


FIG. 6

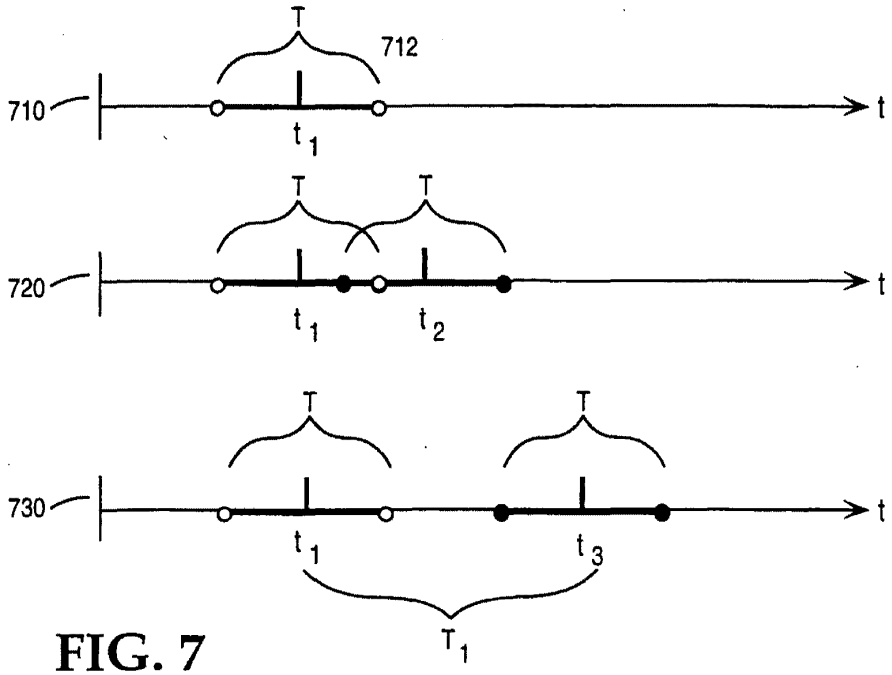


FIG. 7

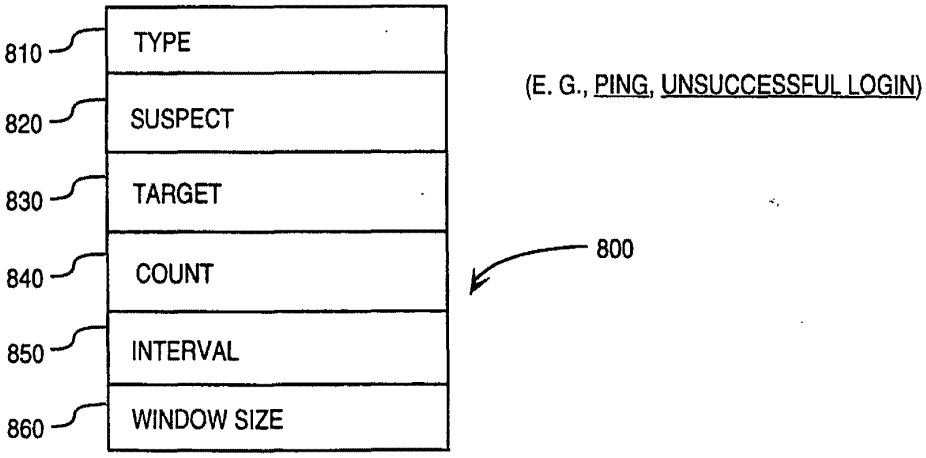


FIG. 8

IPR2024-00027
CrowdStrike Exhibit 1002 Page 702

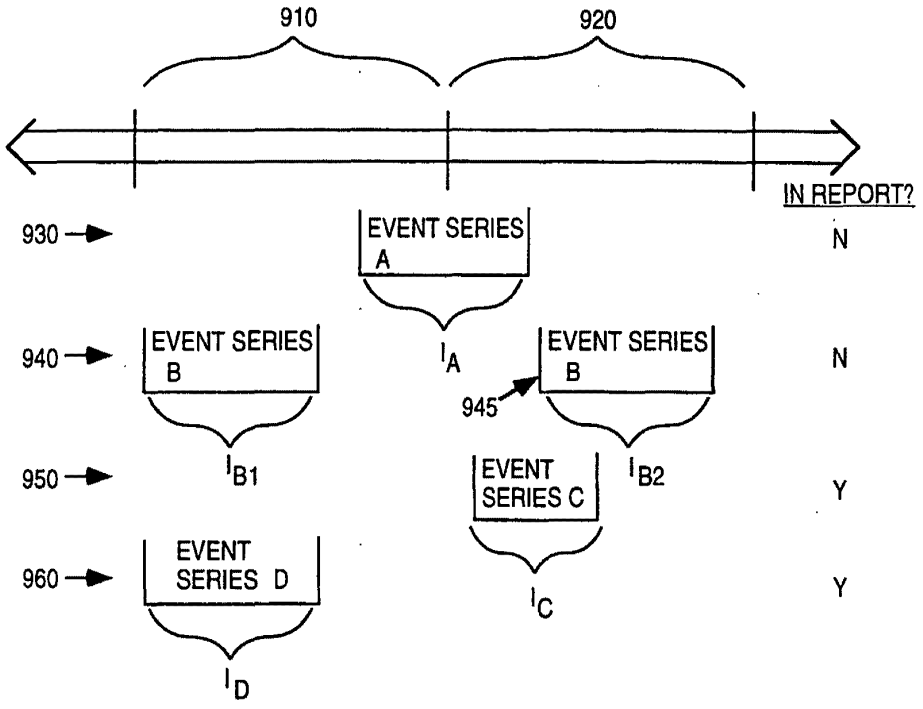


FIG. 9

IPR2024-00027
CrowdStrike Exhibit 1002 Page 703

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/28717

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 11/30, 12/14; H04L 9/00, 9/32
US CL : 713/201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/200; 713/201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searc

Electronic data base consulted during the international search (name of data base and, where practicable, search terms us
BRS
search terms: network, security, login, threshold, ping, alert, attempt

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to cla
X --- Y	US 5,345,595 A (JOHNSON et al.) 06 September 1994, col. 6, line 55 to col. 10, line 56.	1, 18, 26, 31 --- 2-17, 19-25, 30, 33-40
X --- Y	US 5,414,833 A (HERSHEY et al.) 09 May 1995, col. 9, line 24 to col. 16, line 40.	1, 18, 26, 30. --- 2-17, 19-25, 29, 32-40
X --- Y	US 5,278,901 A (SHIEH et al.) 11 January 1994, col. 6, line 60 to col. 16, line 30.	1, 18, 26, 30, --- 2-17, 19-25, 29, 32-40
Y	US 5,475,839 A (WATSON et al.) 12 December 1995, col. 14, lines 11-33.	9, 13

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or prior date and not in conflict with the application but cited to understa the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot l considered novel or cannot be considered to involve an inventive st when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot b considered to involve an inventive step when the document combined with one or more other such documents, such combinatio being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 10 MARCH 2000	Date of mailing of the international search report 26 APR 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ROBERT G. CROCKETT Joni Hill Telephone No. (703) 308-6107

Form PCT/ISA/210 (second sheet)(July 1992)*

IPR2024-00027
CrowdStrike Exhibit 1002 Page 704

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/28717

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim N
X, E ---- Y	US 6,026,442 A (LEWIS et al.) 15 February 2000, col. 3, line 47 to col. 8, line 67.	1, 18, 26, 30, 31 -------- 2-17, 19-25, 27-29, 32-40

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

IPR2024-00027
CrowdStrike Exhibit 1002 Page 705

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 December 2002 (19.12.2002)

PCT

(10) International Publication Number
WO 02/101516 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US02/18981

(22) International Filing Date: 13 June 2002 (13.06.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/298,220 13 June 2001 (13.06.2001) US

(71) Applicant: **INTRUVERT NETWORKS, INC.** [US/US]; 2290 North First Street, Suite 102, San Jose, CA 95131 (US).

(72) Inventors: **GUPTA, Ramesh, M.**; 7073 Hollow Lake Way, San Jose, CA 95120 (US). **JAIN, Parveen, K.**; 2636 Gayley Place, San Jose, CA 95135 (US). **AMIDON, Keith, E.**; 34116 O'neil Terrace, Fremont, CA 94555 (US). **GONG, Fengmin**; 2071 Hall Circle, Livermore, CA 94550 (US). **VISSAMSETTI, Srikant**; 650 Fortuna Common, Fremont, CA 94539 (US). **HAEFFELE, Steve, M.**; 16600 Oleander Avenue, Los Gatos, CA 95032 (US). **RAMAN, Ananth**; 1538 El Oso Drive, San Jose, CA 95129 (US).

(74) Agents: **GALLIANI, William, S.**; Cooley Godward LLP, 3000 El Camino Real, Five Palo Alto Square, Palo Alto, CA 94306-2155 et al. (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR DISTRIBUTED NETWORK SECURITY

(57) Abstract: One aspect of the invention is a method of detecting intrusions on a computer. This method includes storing an intrusion signature describing an attack on a computer. Once a plurality of internet protocol packets is received, the plurality of internet protocol packets collectively containing an information sequence within a series of states, it is rearranged so as to place the information sequence in order. Each state of the series of states is then successively examined so as to correlate the information sequence to the intrusion signature.

WO 02/101516 A2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 706

METHOD AND APPARATUS FOR DISTRIBUTED NETWORK SECURITY

This application claims priority to provisional patent application 60/298,220, which was filed on June 13, 2001.

BRIEF DESCRIPTION OF THE INVENTION

This invention relates generally to computer networks. More particularly, this invention relates to network security sensors and distributed network security sensor architectures used to implement intrusion detection and protection.

BACKGROUND OF THE INVENTION

The prevalence of computer vulnerabilities and malicious computer hackers is well documented. Thus, there are ongoing concerns about computer security. Computer security anxieties span a spectrum of computer configurations, including individual computers, local area networks, and wide area networks.

There are a number of problems associated with current computer security technologies. For example, while there is available information on different computer attacks and countermeasures, there are inadequate techniques for developing, deploying, and managing this information. Another computer security problem relates to the distribution of evolving network security information, such as new computer attack profiles and signatures. It would be highly desirable to provide an efficient and rapid mechanism for distributing this information throughout a network.

As computer network traffic continues to grow, there are increasing demands to improve the processing efficiency of computer security tasks. In order to achieve gigabit and higher intrusion detection speeds, new methods and techniques are required for packet inspection and processing. Ideally, such methods and techniques would be scalable and support dynamic signature set updates.

Another problem with current computer security technologies is that they require a single organization to own, maintain and control their own computer security equipment. It would be highly desirable to allow different organizations to share computer security resources through a subscription-based intrusion detection platform.

Distributed denials of service attacks are a common problem in networked environments. A distributed denial of service attack may take many forms. One common

form of a distributed denial of service attack is for a single computer to send a message to a group of computers instructing the computers to access a target computer. The group of computers then forwards the same message on to a supplemental group of computers. Ultimately, the target computer is inundated with access requests and effectively shuts down. It would be highly desirable to identify a technique for detecting, tracing, and countering distributed denial of service attacks.

In order to provide effective protection for existing computers and computer networks, it is necessary to address these numerous computer security problems. Ideally, a single platform and architecture could be deployed to address these problems. Such a system should be easy to deploy and manage, thereby providing a low cost of ownership. Notwithstanding these cost considerations, the system must have high performance, including the capacity to efficiently detect and protect against known and unknown computer attacks.

SUMMARY OF THE INVENTION

One aspect of the invention is a method of detecting intrusions on a computer. This method includes the step of identifying an internet protocol field range describing fields within internet protocol packets received by a computer. A connectivity range is also established which describes a distribution of network traffic received by the computer. An internet protocol field threshold and a connectivity threshold are then determined from the internet protocol field range and connectivity range, respectively. During the operation of the computer, values are calculated for the internet protocol field range and connectivity range. These values are compared to the internet protocol metric threshold and connectivity metric threshold so as to identify an intrusion on the computer.

Another aspect of the invention is a method of detecting intrusions on a computer. This method includes storing an intrusion signature describing an attack on a computer. Once a plurality of internet protocol packets is received, the plurality of internet protocol packets collectively containing an information sequence within a series of states, it is rearranged so as to place the information sequence in order. Each state of the series of states is then successively examined so as to correlate the information sequence to the intrusion signature.

Another aspect of the invention is a method of forming a virtual intrusion detection system. This method includes the step of positioning a set of sensors in a network

IPR2024-00027
CrowdStrike Exhibit 1002 Page 708

environment, each sensor supporting multiple logical traffic paths. The method also includes the step of providing a set of sensor management systems corresponding to the set of sensors. This set of sensor management systems enforces a set of virtual intrusion detection systems, wherein each virtual intrusion detection system corresponds to a predetermined logical traffic path through the set of sensors, each virtual detection system providing sensor traffic information solely to authorized parties.

Another aspect of the invention is a method of provisioning a computer against computer attacks. This method includes constructing a hierarchy characterizing different computer attacks and counter measures, and traversing this hierarchy to identify computer attacks and countermeasures relevant to a target platform. Detection and protection measures are collected in response to this traversing. These detection and protection measures are then downloaded to a security sensor associated with the target platform.

The invention provides a single platform and architecture to address a variety of network security problems. The system of the invention is easy to deploy and manage, and thereby provides a low cost of ownership. However, the system of the invention also has high performance, including the capacity to efficiently detect and protect against known and unknown computer attacks.

BRIEF DESCRIPTION OF THE FIGURES

The invention is more fully appreciated in connection with the following detailed description taken in conjunction with the accompanying drawings, in which:

FIGURE 1 illustrates a computer network environment implementing the network security techniques of the invention.

FIGURE 2 illustrates a network security sensor implemented in accordance with an embodiment of the invention.

FIGURE 3 illustrates processing steps performed by an embodiment of the network security sensor of the invention.

FIGURE 4A illustrates an embodiment of a hardware sensor management module utilized in accordance with an embodiment of the invention.

FIGURE 4B illustrates a specific hardware implementation of a network security sensor of the invention.

FIGURE 5 illustrates input and output information associated with the protocol parsing state machine of the invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 709

FIGURE 6 illustrates general state machine processing operations performed in accordance with an embodiment of the invention.

FIGURE 7 illustrates a state machine transition table implemented in accordance with an embodiment of the invention.

FIGURE 8 illustrates a transition operation specification format utilized in accordance with an embodiment of the invention.

FIGURE 9 illustrates a signature processing architecture utilized in accordance with an embodiment of the invention.

FIGURE 10 illustrates an example of state machine processing performed in accordance with an embodiment of the invention.

FIGURES 11A & 11B illustrate exemplary state machine instructions to be carried out in accordance with an embodiment of the invention.

FIGURE 12 illustrates an exemplary hardware configuration for carrying out signature processing operations in accordance with an embodiment of the invention.

FIGURE 13 illustrates a sensor control system utilized in accordance with an embodiment of the invention.

FIGURE 14 illustrates a global sensor management system implemented in accordance with an embodiment of the invention.

FIGURE 15 illustrates an update server implemented in accordance with an embodiment of the invention.

FIGURE 16 illustrates a hierarchical attack categorization structure constructed and utilized in accordance with an embodiment of the invention.

FIGURE 17 illustrates processing steps performed in accordance with a hierarchical attack categorization process utilized in accordance with an embodiment of the invention.

Like reference numerals refer to corresponding parts throughout the several views of the drawings.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 illustrates an exemplary computer network 20 incorporating network security devices and processes associated with the invention. The network 20 includes a set of network security sensors 22 configured in accordance with the invention. Each sensor 22 operates as a platform to implement local and distributed security operations performed in accordance with the invention. Figure 1 illustrates a set of primary sensors 22 and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 710

redundant sensors 24. Preferably, a dedicated link 25 is positioned between the primary sensors 22 and the redundant sensors 24. As discussed below, the primary sensor 22 updates the redundant sensor 24 with changes in the configuration data. This ensures that the primary sensor 22 and the redundant sensor 24 are synchronized and that the redundant sensor 24 can be activated in the event of the failure of the primary sensor 22.

A sensor management system 26 is associated with a sensor 22 or set of sensors 22 and 24. The sensor management system provides supervisory control of a sensor 22. The sensor management system 26 may be used to implement a shared-resource virtual intrusion detection system, as discussed below. A single sensor management system 26 may be used to control multiple sets of primary sensors 22 and redundant sensors 24.

The combination of the sensor 22, redundant sensor 24, and sensor management system 26 is referred to as a local sensor security module 27. As shown in Figure 1, local sensor security modules 27 may be distributed throughout a network. In this example, local sensor security modules 27_1 through 27_N are positioned between an enterprise network 30 and Internet service providers 28_1 through 28_N. In addition, a local sensor security module 27_0 is positioned between the enterprise network 30 and a protected server 32.

The operations of the local sensor security modules 27 may be coordinated through a global sensor management system 34. The global sensor management system 34 performs distributed system management operations and provides a global consolidated view of all sensors and all the traffic these sensors are monitoring. In addition, the global sensor management system 34 supports the implementation of a global shared-resource virtual intrusion detection system. In addition, the global sensor management system 34 tracks information from the local sensor security modules 22 to identify and respond to distributed denial of service attacks.

Figure 1 illustrates that the network 20 includes an update server 38. The update server 38 is used to coordinate the delivery of signature and software updates to the local sensor security modules 27, as discussed below. Preferably, the update server 38 is protected by a firewall 36.

The overall architecture of an embodiment of the invention has been described. Attention is now directed toward a more particular description of the individual components of the architecture.

Figure 2 illustrates a sensor 22 configured in accordance with an embodiment of the invention. Preferably, the sensor 22 includes a set of processor 40_1 through 40_N, with

IPR2024-00027
CrowdStrike Exhibit 1002 Page 711

each processor optimized to perform a different function, as discussed below. The processors 40 are connected to a system bus 42 or a set of buses or a switching fabric, which are represented by the single system bus 42. Also connected to the system bus 42 is a set of input/output ports 44. The ports 44 provide interfaces for routing network traffic. In addition, they include interfaces for the sensor management system 26.

In one configuration, the system bus 42 is also connected to a memory 50, which includes primary and/or secondary memory. The memory 50 stores a set of executable programs utilized to implement functions of the invention. In an alternate embodiment of the invention, the executable programs are stored in memory associated with each processor that executes a program.

In the embodiment of Figure 2, the memory 50 stores a sensor management module 52, which coordinates overall sensor operations. Alternately, the sensor management module 52 may be implemented in a separate processor or processor board used to coordinate overall sensor operations. In the embodiment of Figure 2, the memory 50 stores a response module 54 for coordinating responses to processing exceptions.

A packet decoder 56 is also stored in memory 50. The packet decoder 56 coordinates the decoding of network packets and performs protocol conformance verification. Alternately, the functionality of the packet decoder 56 is implemented with a dedicated processor 40. A load balancer 58 is preferably used to distribute processing responsibilities across the processors 40.

Figure 2 also illustrates a statistical analysis and distributed denial of service detection module 60. This executable program analyzes statistical patterns associated with processed traffic. In addition, it identifies distributed denial of service attacks. A path marker insertion module 61 is used to insert path markers into network traffic. The path markers are used to identify the actual path traversed by the distributed denial of service attacks, as discussed below.

The memory 50 also stores an anomaly detector 62, which is used to identify network traffic anomalies indicative of an attack. A fixed-field detector 63 and a protocol parser 64 are also stored in the memory 50. The protocol parser 64 is implemented with a set of state machines 66 and associated tables 67. As discussed below, the state machines process a data stream by generating intrusion detection information with each state transition. In combination, the fixed-field detector 63 and the protocol parser 64 operate as a signature processing system, as discussed below. Supporting this signature processing

IPR2024-00027
CrowdStrike Exhibit 1002 Page 712

system is a stream orderer 51 that organizes data streams for a token detector 53. In turn, this token detector 53 transmits tokens containing state information and other instructions for the protocol parser 64.

The memory 50 also stores a classification and pattern-matching module 68. This module has an associated set of intrusion signatures 70. The module is used to compare incoming network traffic with the set of intrusion signatures 70, as discussed below.

The memory 50 also stores an encrypted session monitoring module 72. This module 72 allows the sensor 22 to decrypt otherwise secure network traffic in a non-intrusive manner. As discussed below, protected key information 76 stored within the sensor 22 is used to implement the operations performed by the encrypted session monitoring module 72.

The memory 50 also stores a data stream processor 74. The data stream processor 74 reassembles IP fragments and sends the reassembled IP fragments back to the load balancer. In addition, the data stream processor 74 reassembles TCP streams and forwards the reassembled streams to the signature and anomaly detection module 62.

The memory 50 also stores a fail-over switch module 78. The fail-over switch module 78 is used to synchronize information between a primary sensor 22 and a redundant sensor 24 and to switch control from the primary sensor 22 to the redundant sensor 24 in the event that the primary sensor 22 fails.

The processing performed by the sensor 22 is more fully appreciated in connection with Figure 3. Figure 3 illustrates processing steps performed by the sensor 22. Incoming traffic to the sensor 22 is processed for packet decoding, protocol conformance verification and load balancing 81. The packet decoder 56 and load balancer 58 are used for this operation.

The traffic is then processed for statistical analysis and distributed denial of service detection 82. These operations may be performed by module 60. Stream processing is then performed 83. This operation may be implemented with the data stream processor 74. Signature and anomaly detection 84 is then performed. The anomaly detector 62 may perform these anomaly detection operations.

The overall processing is supervised through sensor management 86, which is implemented with the sensor management module 52. A response module handles response processing 85. In particular, the response processor 54 determines the response actions for

IPR2024-00027
CrowdStrike Exhibit 1002 Page 713

the specific attack. The response processor 54 is configured by the system administrator. Once configured, the response processor 54 responds to specific attacks.

As previously indicated, the sensor 22 may be implemented with a set of processors. The different software modules stored in memory 50 run on selected processors of the set of processors. Thus, for example, the sensor management module 52 has been implemented to run on a x86 single board computer, an example of which is illustrated in Figure 4A. The packet decoder and load balancer have been implemented to run on two network processors (Sitera Prism IQ2000). The response processor 54 has been implemented to run on a set of high performance CPUs (e.g., the SiByte Mercurian SB-1250). The statistical analysis and distributed denial of service (DDOS) detection module has been implemented to run on a co-processor (e.g., the FastChip PolicyEdge processor). The classification and pattern-matching module has been implemented with a co-processor (e.g., the Switch-On PM2329). The anomaly detector 62 has been implemented to run on a set of high performance CPUs (e.g., the Sitera Prism Connect 821308). Figure 4B illustrates a specific circuit topology used to implement an embodiment of the sensor 22. In this embodiment, software modules executed by a processor are stored in the primary memory associated with the processor.

Returning to Figure 3, after packet decoding and load balancing 80, statistical analysis and DDOS detection 84 is performed. The statistical analysis and DDOS detection module 60 operates in connection with a path marker insertion module 61. The path marker insertion module 61 inserts DDOS identification information into the network traffic processed by the sensor 22. The module 60 also monitors the DDOS identification information received from other upstream sensors in the network. When violations of DDOS detection profiles are observed, appropriate DDOS attack flags are set. This can result in remedial action performed at the sensor 22. In addition, the attack flag signal is transported across the network to the protected computer 32, which takes additional remedial actions to prevent the DDOS attack. Various techniques for implementing these operations are discussed below.

Returning to Figure 2, the anomaly detector 62 is used to identify computer attacks. In this context, anomaly means any event, state, or behavior that is considered to be abnormal by certain pre-defined standards. For example, the existence of a remote root shell on a system that only expected console root login is an anomaly. Seeing large numbers of successive small IP fragments is an anomaly. A web server suddenly seeing a lot of requests from new IP addresses may also be considered an anomaly.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 714

Anomaly-based intrusion detection approaches complement the signature-based approaches by offering a means to detect attacks whose signatures are not yet known or attacks that exhibit modified behavior (e.g., intentionally stealthy attacks or variants of existing attacks in new environments). The term system refers to any entity whose relevant state or behavior is under observation. For example, it can be a host or server, a given network application, or a person. The anomaly detector 62 is typically implemented in accordance with a number of operations. First, measures (or observations) of normalcy are defined for a given system. Next, a characterization of the normalcy of the system is created. This characterization is generally in a form of distributions for these measures or their derivatives. This may require a learning or training process. Next, an algorithm for building a run-time characterization of the system is defined. Measures of discrepancy between the normalcy and the run-time characterization are then defined. Once again, this may require learning or training. The measure of discrepancy and the way the actual measurement is obtained can introduce inherent differences that are accounted for in the threshold determination step. Finally, anomaly thresholds for generating appropriate alarms are defined. This approach can be implemented using multiple techniques, including statistical, neural nets, and other forms of learning mechanisms.

The anomaly detector 62 creates a characterization of the normal behavior of the system in order to achieve accurate anomaly detection (i.e., with low false positive and low false negative rates). Since different systems have different behaviors, a new characterization needs to be created for each new system to be protected through anomaly detection. In one embodiment of the invention, the anomaly detector 62 operates in two phases. In a training phase the target system needs to be in an attack-free state. Depending on the resource availability, training can be conducted either online or offline. In the online case, training data comes directly from the real-time traffic captured while the system is in operation. In the offline case, training data comes from previously captured traffic traces, which are stored in a file. The length of the training phase will typically depend on the inherent variability of the system. Training can stop automatically when certain stability criteria have been met. However, the user should be able to turn on the training mode at any time.

After the conclusion of training, the anomaly detector 62 operates in a detection phase. The detection phase produces anomaly scores for the observed packets based on the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 715

characteristic similarity between the observed and normal profile. A higher score will indicate a higher degree of deviation from the normalcy and thus a stronger intrusion alert.

While the training accounts for the difference in characteristics from system to system, there is also variability in time (e.g., the time of day) that may be significant enough to require new profiles for effective detection. The anomaly detection module supports the following general means for adaptation. First, an interface for human analysts is supplied to allow the input of final alert assessment results and to keep track of the false alarm rate changes. In the case where the false alarm rate increases and stays at a higher level, this is a good indication of a system/environment change that can be accounted for by re-training the anomaly detector 62. In the case where false alarm rates fluctuate periodically with time, it is a good indication that a new set of profiles with a different periodicity is required.

Another adaptive technique that can be implemented by the anomaly detection module 62 is to support multiple profiles that can be dynamically updated with time, or equivalently one profile that adapts continuously but more quickly. To better support creation of new profiles dynamically, the anomalous packets should be kept in a log file until it is determined that they were normal, or to be moved to long-term archive. At that time, these logged packets can be used to create the new profiles or to re-train existing profiles.

The anomaly detector 62 has been implemented to detect two types of anomalies. The first type of anomaly is identified based upon a normal profile of non-attack Internet packets. This method helps detect those attacks that are realized through specially crafted packets or other attack packets, such as denial of service or DDOS attacks. The second type of anomaly is identified based upon the normal traffic profile of a target domain, which may be a single host/sever, a sub-net, or an enterprise network. The detection is based on the change of traffic patterns over network links.

The first technique of profiling typical non-attack packets relies upon the occurrence or co-occurrence of values in a selected set of fields. That is, in the absence of active attacks, there are generally defined patterns or ranges of values taken by the header fields of a packet. These patterns can be identified through statistical analysis or learned by artificial neural networks. These patterns can then be compared against the actual field values of a packet on the wire to detect abnormal packets. In one embodiment, this comparison is carried out by establishing a threshold at one extreme of the range or pattern in question, and checking to see if a packet's field value exceeds this threshold. In addition, some

IPR2024-00027
CrowdStrike Exhibit 1002 Page 716

"forbidden" rules are manually introduced to ensure that certain packets are always flagged due to their potential damaging impact. For example, a ping packet (ICMP ECHO REQ) with multicast/broadcast destinations is a cause for concern, and is thus an anomaly.

Advantageously, such normal profiles can be created from packet traces generated entirely from known non-attack implementations of the protocols. Thus, it is not necessary to learn the profiles during a guaranteed attack free session.

By way of example, three types of packets can be characterized: TCP, UDP, and ICMP. In the case of TCP and UDP packets, one embodiment of the invention establishes normal profiles characterizing the contents of (or, in some cases, simply checking certain values of) one or more packet fields. The first of such fields, which can also be thought of as variables, is IP packet length. This embodiment of the invention establishes a normal profile, or range, of packet lengths. Any packets with lengths outside this range, or beyond some upper or lower threshold, can be considered abnormal and marked as anomalous. Those of skill in the art will recognize that the invention includes the determination of thresholds from such ranges through any known means, for example choosing a percentile value of the range.

The next variable concerns fragment bits. In the case where the "don't fragment" bit is set, one expects a corresponding offset equal to zero. If the "more fragment" bit is set, all fragments except the last should have offset values from a finite set, as determined by MTUs of the known network switches/routers. These offsets should also never be more than the maximum IP datagram length. A violation of any of these conditions can be considered as an anomaly. A related variable concerns the IP fragmentation identifier field. If multiple packets with overlapping fragmentation offsets are detected, this condition is also considered an anomaly.

Another variable compares the protocol identification field to the finite set of known IP protocols; if the new protocol identifier is not recognized, this signals an anomaly. Yet another variable examines the set of TCP code bits to determine whether they contain legal bit combinations. Combinations other than legal ones signal an anomaly. For example, OS scanners are known to use strange combinations of code bits (e.g., SYN and FIN) to determine the target OS type.

One variable specific to UDP concerns protocol data unit (PDU) length. A normal range for the PDU length of UDP packets can be established, with packets exceeding this range, or a threshold at the upper end of the range, being marked as anomalous. The set of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 717

configured services (e.g., VOIP or other packet audio/video applications) can assist in establishing initial values for such a range, as various applications typically transmit UDP packets with characteristic PDU length ranges.

A range can also be established characterizing the various destination addresses of IP packets received. Here, the range can characterize the destination address in any known manner. For instance, the range can be a numerical range characterizing the numerical values of the various destination addresses. It can also be a range of address/mask pairs. Depending on the size of the secured domain, an accurate profile of destination addresses may require capturing multiple networks and/or subnets. A different yet related range characterizes the range of IP packet destination ports. Here, for instance, the finite set of ports corresponding to legally configured services can be characterized. Packets with destination ports outside this set can be designated as anomalous.

With respect to ICMP packets, a different set of variables may be considered. As in the TCP/UDP case, IP packet length can be profiled, as there is a well-defined set of known ICMP messages that can be used to define a normal range. As this range is different from TCP/UDP packets, ICMP packet length can be maintained as a separate variable from TCP/UDP packet length. Another variable concerns packet fragmentation. This fragmentation can simply be profiled as in the TCP/UDP case above or, because fragmentation in ICMP packets is rare, its occurrence can automatically be considered as an anomaly. For example, large pings on limited occasions can generate fragmentation, and are also often a cause for alarm.

The range of protocol identifications can also be considered a variable and characterized as above. Finally, the range of ICMP type and code combinations can be characterized, as there exists a well-defined set of ICMP type and ICMP code combinations. Any combinations outside this set can therefore be considered anomalous. In addition, a given domain is made aware of the transmission of an ICMP redirect packet by one of its routers. This behavior can also be considered as outside this range, and thus anomalous.

It should be noted that various embodiments of the invention simply include profile generation without limiting themselves to generating profiles by any particular method. Indeed, any method can be used. For example, variables such as IP packet length can be characterized statistically by sampling a significant number of packets. As another example, a novel approach to profiling data rates is also disclosed in a co-pending U.S. patent application entitled "Method and Apparatus for Detecting a Distributed Denial of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 718

Service Attack," filed on June 13, 2002 with docket no. ITRU – 001/00US, hereby incorporated in its entirety. This approach can be extrapolated to variables besides data rate, and also provides a method of updating such a characterization.

The second anomaly detection technique of traffic profiling characterizes the normal behavior of the target system. This behavior can be termed connectivity behavior, as it describes traffic flowing over the target system's connection to a network. The target system can be a whole enterprise network, an individual server or host machine, a class of services on a machine, or a particular user on a machine. Packet length and packet rate are the two most basic measures on a link.

In order to use these two measures to more effectively characterize the normal connectivity profile, the basic measures can be further broken down according to several attributes. For instance, packet length and packet rate can be categorized or profiled according to external link group. Where a domain is connected through multiple parallel links to the Internet or another network, inherent factors such as physical link speed and MTUs for the networking devices in each link group all affect packet rate and packet length for each link. Similarly, dynamic and asymmetric routing makes it likely that packets between a given pair of hosts may actually take different paths, or links, at various times. Aggregating traffic across all links in the link group thus provides a more complete traffic profile.

Packet length and packet rate can also be profiled according to time of day. Time of day is known to affect the type of networking applications, and thus traffic levels, invoked over a network link. For example, enterprise applications often schedule automated batch processing at after-hours times, raising network traffic. Time of day thus has an effect on both packet rate and length distribution on such links, so such a breakdown helps reduce the complexity of the normal profile with respect to time.

Basic measures can also be profiled according to packet destination. As different destinations generally offer different protocols and applications, utilize different service platforms, and have different network connectivity attributes, their basic traffic measures will vary. Subdividing basic measures according to destination thus represents further profile refinement. The same reasoning also supports the profiling of basic measures by source.

Such basic measures can also be subdivided according to service, i.e., application protocol. For example, packet length and rate profiles can be made specific to the http, ftp,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 719

or telnet protocols. Those of skill in the art will realize that not only do these protocols have very different but stable profiles in terms of measures such as PDU length, but they also exhibit very different packet rates (or, equivalently, packet arrival intervals) depending on whether human interaction is involved. Per protocol measures are thus useful in distinguishing among human users, client/server processes, or tunneling Trojans.

Depending on the domain to be protected and the categories of attacks to be protected against, detailed measures from one or more of the above categories may be appropriate, all combinations and permutations of which are encompassed by the invention. Practicing the invention thus allows users the choice of profiling network traffic according to any of the above fields or ranges, further subdivided by any of the aforementioned subdivisions. Flexibility in choosing such fields and subdivisions thereof also affords users maximum attack protection within their limited resources. Different destinations generally offer different kinds of service (thus a different portfolio of network protocols and applications). Different destinations may be using different service platforms and may have different network connectivity (different transport networks, different link bandwidths). These all have an impact on the basic traffic measures. However, when the destinations are remote domains (outgoing traffic), keeping per destination measures may be too expensive due to potentially large number of destination hosts and sub-nets. It may be reasonable for extra-net types of deployment. Again, per destination breakdown represents further refinement of profile measures.

The same reasoning applies for keeping per source measures. Similarly, per source measures may be too expensive to keep for incoming traffic, except for extra-net deployments. Assuming that one is expecting denial of service attacks from outside and that the sources are easily spoofed, refinement of measures based on source address is not very desirable.

The anomaly detector 62 also identifies application anomalies through system connectivity profile checking. Two types of connectivity profiles have been identified to capture the normal connectivity behavior of the target computer. The first type consists of a set of activity vectors. It is aimed at detecting general changes in the connectivity pattern to or from the target system/domain. The following vectors may be defined: incoming protocol, outgoing protocol, incoming destination port seen, incoming source port seen, incoming destination IP address seen, incoming source IP address seen, outgoing

IPR2024-00027
CrowdStrike Exhibit 1002 Page 720

destination port seen, outgoing source port seen, outgoing destination IP address seen, and outgoing source IP address seen.

Take the incoming protocol vector, IN = < p1, p2, p3, p4, p5, p6 > as an example. Pi can be TCP, UDP, ICMP, HTTP, FTP, or DNS. The elements in the vector (Pi's) are ordered from the most often seen to the least often seen protocols. Basic statistics are collected on the number of packets seen that are of these protocol types or carrying those types. Although the update of the vector can be as frequent as one for each packet arrival, the comparison of the run-time vector to the normal profile will typically happen at larger time intervals.

The ordering of the elements in the vector can be based on a number of different criteria, for example, the number of packets related to it, the total number of bytes related to it, length of the flow/session related to it, number of fragmented packets related to it, etc. In general, each element in the vector represents a separate security event and our profile captures the distribution of this set of events.

The length of the vector can be adjusted based on what measure is being used and at what granularity. For example, active incoming destination IP address seen vector can be based on network prefix or sub-net address, depending on which level the deployment environment is at (top level ISP versus local access ISP). There are only four classes of Ipv4 addresses currently used but there will be a lot more network prefix and even more sub-nets.

For the cases where the range of possible values is large, it is reasonable to define the vector with a small length N so that the first N-1 elements capture the most active specific category while the last element captures the aggregate of all the rest.

Another form of application anomaly that is identified by the anomaly detector 62 is the request-response anomaly. There are a significant number of denial of service attacks that exploit the fact that the design and or implementation of many network applications do not handle exceptions well during operation. For example, CVE-1999-0274 documents a denial of service attack on NT DNS, which is based on sending a response to a query that was never made. CVE-2000-0041 documents a Macintosh vulnerability where the Macintosh generates a large number of ICMP data-grams in response to malformed data-grams. This makes the Macintosh a packet amplifier for flood attacks.

By identifying the set of commonly used request-response packet pairs and creating a simple balance profile, the anomaly detector 62 detects most attacks. This is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 721

accomplished by: (1) keeping a run-time count of request and response packets separately, (2) establishing a rate profile for the occurrence of these individual packets and generate an alert if the threshold of deviation is crossed, and then (3) correlate the request and response by simply checking the balances. Request-response examples include: TCP SYN-TCP SYN & ACK; TCP FIN – TCP FIN & ACK; CIMP ECHO REQ – ICMP ECHO REPLY, ARP request – ARP response, DNS query –query response, HTTP request – response.

In order to support these profiles, statistics must be collected on protocol fields beyond the TCP/UDP/IP header. Bi-directional traffic needs to be examined and correlated based on information fields that are dependent upon the application protocol.

The anomaly detector 62 is one mechanism utilized by the invention to prevent attacks. Another mechanism utilized in accordance with the invention is signature processing. The signature processing system detects attacks by looking for signatures that are specific combinations of patterns (e.g., numerical field values, string matches, and the like) existing in monitored network traffic.

In one embodiment of the invention, signature processing is implemented with the fixed-field detector 63 and the protocol processor 64. The fixed-field detector 63 performs per-packet analysis of protocols in which numeric fields can be located by simple offset/mask specifications. It is not intended to process protocols that require string matching or state to be kept between packets.

The protocol processor 64 performs complex protocol processing for protocols requiring string matching or state maintenance for up to the duration of a flow. Preferably, the protocol processor 64 has an associated functionality that ensures that all packets for a particular flow are presented to the next stage in the system in order, and with enough overlapping packet information to ensure detection of matches crossing packet boundaries. An associated token detector is preferably used to examine an in-order stream of packets for string matches indicating protocol delimiters or attacks. The token detector presents these in order of occurrence to the protocol parsing state machine 66 as tokens indicating the string that matched along with the position and length of the match.

Both the fixed-field detector and the protocol parser 64 generate Signature Sub-IDs (SSIDs) that indicate the presence of subsets of the patterns required for detection of a particular signature. The classification and pattern matching module 68 accepts the SSIDs in the order in which they occur and determines if all the SSIDs required to form the complete pattern of a signature occurred. The pattern signatures are stored as intrusion

IPR2024-00027
CrowdStrike Exhibit 1002 Page 722

signatures 70. For each complete signature, the classification and pattern matching module 68 raises an AttackID (AID) that indicates the attack that was found in the communications being monitored.

The response processor 54 responds to AIDs. The response processor 54 generates alerts that provide notification of an attack. The response processor 54 also attempts to prevent the attack. Short-term responses include terminating TCP connections. Long-term responses include packet logging for further analysis to improve detection and response.

The protocol parser 64 is implemented using a generic state machine that is configured for each protocol parsed. The generic state machine allows changes to protocol parsing without the introduction of new code onto the sensor. In signature detection, where only a subset of the fields of any given protocol is known to have vulnerabilities in various implementations, this allows performance optimization without limiting the future detection capability of the sensor.

Those skilled in the art will appreciate that the generic state machine approach of the invention has a number of other advantages. For example, this approach allows the separation of signature development and code development, allowing intrusion detection specialists to do the former, while software engineers focus on the latter, with fewer dependencies between them. This approach also facilitates improved customization, since the sensor management system 26 can customize the protocol parsing mix per sensor through the state machine configuration tables rather than through different code images. The generic state machine approach also facilitates rapid implementations to support new protocols.

The protocol parser 64 allows one to specify the parsing of application layer protocols in a table-driven manner. This allows new or updated parsing to be introduced without requiring any new code (e.g., a new software image), which in turn allows real-time dynamic updates of the entire signature detection engine without interruption to current processing.

The protocol parser 64 is implemented to use generic operations executed on each state transition in order to collect information from the data stream carrying the protocol under analysis. The collected information is stored and is subsequently used in order to modify the future behavior of the state machine.

Figure 5 illustrates the inputs and outputs for the state machine 66. The inputs that drive the protocol parsing state machine include the current state, the configured state

IPR2024-00027
CrowdStrike Exhibit 1002 Page 723

transition table, the configured field comparison tables, and information derived from the token detection engine 69, which includes information derived from the in-order application layer data sent between hosts during a monitored connection. The derived information includes a stream of detected string matches in the data (tokens), information on the current progress of the connection between the hosts (connection state), and the set of initially empty generic data locations that are used during processing (data locations).

The tokens detected within the data stream are provided to the state machine in order from the beginning of the stream. Since the sensor only sees the stream as a set of inter-related non-overlapping, possibly out-of-order packets, the token matching system must order the data and ensure that matches are made across packet boundaries. Provided to the state machine along with the token are the offset of the token in the packet in which it was found, and the length of the token match. In addition to the string match tokens, the following protocol tokens are given to the state machine in protocol events: start of processing of packet from client to server, start of processing of packet from server to client, processing reached end of packet from client to server, processing reached end of packet from server to client, close of connection from client to server, and close of connection from server to client. The connection information includes: the stream offset of the beginning of the packet being processed, which allows calculation of stream offsets of token matches; stream offset transition triggers; and a single stream processing suspend offset, which can be set to –1 to indicate processing should be suspended for the remainder of the stream. The data locations all start out cleared to zero when a new instance of a state machine is created. During processing, the generic state machine can modify the values of data in these locations and makes tests on them.

At the highest level, the state machine output is a Boolean value indicating if an attack signature (Signature SubIds) has been found in the application data sent on a monitored connection up to the time of processing. If no Signature SubIds was found, no other information is needed or given. If one was found, the following information may be provided: the current state, the token being processed, including packet offset, the transition operation specification that was being processed, the packet being processed, and the state of the data locations.

The general operation of the state machine is illustrated in Figure 6. There is an initial wait for a token (block 87). Once received, the token is delivered to the state machine (block 88). The transition operation pointer is then set to the first transition

IPR2024-00027
CrowdStrike Exhibit 1002 Page 724

operation for the token (block 89). The transition operation is then loaded (block 90) by referring to the transition table (block 91). The transition operation is then executed (block 92) with reference to the current state (block 93) and the data locations (block 94). If an SSID is found (block 95 – Yes), it is delivered to an attack detector (e.g., the classification and pattern matching module 68) for processing (block 96). If an SSID is not found (block 95 – No), a decision is made regarding whether the flow is completed (block 97). If the flow is completed (block 97 – Yes), then processing is done. If the flow is not completed (block 97 – No), a decision is made regarding whether there are more transition operations (block 98). If there are more transition operations (block 98 – Yes), the transition operation pointer is set to the next transition operation (block 99) and a transition operation is loaded (block 90). If no more transition operations exist (block 98 – No), there is a wait for the next token (block 87).

The state machine 66 may rely upon field comparison tables that specify comparisons and corresponding signature subIDs for the various types of comparisons that can be done on protocol fields. The default comparison is a sub-string search, which does not require any field comparison table, since it is implicitly carried out by the token matching engine. Exact match and matches anchored at the start and end of a field are handled as special cases of this mechanism with operations in the transition table, an example of which is provided below. Comparisons for specific values, ranges of values, and regular expressions require field comparison tables. Each field for which a comparison of a type is required will have its own comparison table referred to by a sensor tag that starts the search.

Example field comparison tables utilized in accordance with an embodiment of the invention include a numerical equal field comparison table, a numerical range field comparison table, and a regular expression field comparison table. In one embodiment, the numerical equal field comparison has the following columns: match value (unsigned 32 bit integer) and SSID (Signature SubID, an unsigned 32 bit integer). In one embodiment, the numerical range field comparison table has: a low value (unsigned 32 bit integer), a high value (unsigned 32 bit integer), and an SSID (Signature SubID, an unsigned 32 bit integer). In one embodiment, the regular expression field comparison table includes: a regular expression (null-terminated string) and an SSID (Signature SubID, an unsigned 32 bit integer).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 725

An example of the transition table format is provided in Figure 7. The transition table is a two-dimensional array where the dimensions are the allowable states and the set of detectable tokens. Each location in the array contains a list of one to n transition operation specifications, which are actions to be taken when the transition is activated. The last transition operation in the list either switches to a new state or the state machine remains in the current state.

Transition operations fall into four basic categories: (1) change state; (2) modify some value in the data locations; (3) test against the values in the data locations, signaling for further action when certain results are obtained; and (4) compare field values. In one embodiment, each transition operation specification is a 64-bit value configured as shown in Figure 8. The 16 least significant bits are the operation code indicating the operations to be run. Each operation uses some combination of the remaining fields as operands.

Signature processing is more easily understood with reference to the following example, which illustrates the payload of an exemplary IP packet and the processing steps involved in parsing it. Assume that a sensor 22 receives a packet containing the following http GET request in its body:

GET /dir1/dir2/dvwssr.dll ? Param1 = value1 & Param2 = value2 HTTP 1.0

Figure 9 illustrates an architecture for processing this packet in accordance with an embodiment of the invention. Once the packet is received by the sensor 22, it is cached while its contents are analyzed. The packet is first analyzed by the fixed-field detector 63, which examines both fields within the packet header and fixed locations within the packet payload. Conceptually, the fixed-field detector 63 scans the packet for attacks that leave characteristic traces in certain specific fields. One example is the LAND attack, which consists of streams of TCP SYN packets whose source IP address and TCP port number are set equal to the destination IP address and TCP port number, respectively. The fixed-field detector 63 thus scans the source and destination address fields, which always occur in fixed or predictable locations, and transmits an SSID if they are equal.

The stream orderer 51 simultaneously arranges the contents of IP packets in their proper order, so that attacks spanning multiple packet payloads can be detected. The token detector 53 parses the packets including their ordered streams of data, searches for traces of an attack, then transmits tokens summarizing its findings and instructing the protocol parser

IPR2024-00027
CrowdStrike Exhibit 1002 Page 726

64 accordingly. The token detector 53 scans the contents of each field (e.g., the various header fields, and then the body), compares each to a set of tables containing strings indicating portions of an attack, and transmits any matches in the form of tokens to the protocol parser 64. In this manner, the token detector 53 notifies the protocol parser 64 of strings that match an attack signature, what field or portion thereof (i.e., what state) the string is located in, and any instructions for manipulating the packet contents. The protocol parser next 64 executes the token instructions to determine whether any SSIDs exist and, if so, passes them on to the attack detector 55. If the attack detector 55 assembles a complete set of SSIDs indicating an attack, the appropriate AID is passed to the response module 54, which can discard the packet from cache, raise an alert, or take other remedial actions. If no attack is detected, the attack detector 55 can simply pass the packet on from the cache to its final destination.

The operation of the protocol parser 64 is more easily understood in connection with the http GET request shown above. This request is an http-protocol request to get the file dvwssr.dll, a restricted-access file. A GET request for this file is one signature of an attack. Upon parsing the packet(s) containing this statement, the token detector 53 compares its contents to tabulated strings, some of which (as this is a well-known form of attack) includes the GET request. The token detector 53 thus sends a token indicating the presence of a GET request and the presence of the string "dvwssr.dll" in the URI field. It also sends tokens indicating transitions to the next state. Here, for instance, it sends a token indicating the presence of a "?" denoting a state transition to the beginning of a parameter field, and a token indicating the "&" character for transitioning state to the next parameter field. Also sent is a token indicating that the payload is written in protocol http 1.0.

Figure 10 illustrates the protocol parser 64 acting on the tokens it receives. As mentioned previously, the protocol parser 64 contains a number of generalized states, shown as circles in Figure 10. It moves among these states, and executes instructions, according to tokens it receives indicating the presence of various string matches, which are shown as boxes in Figure 10. At each state, the protocol parser 64 executes various instructions to manipulate the information within that state if necessary, and compare it with attack information stored in the tables 67.

In the present example, the protocol parser 64 would begin at the start circle. Upon receiving the GET token, it would proceed to the URI state and compare the GET and "dvwssr.dll" tokens to its list of tabulated attacks, which may or may not result in the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 727

generation of an SSID. Note here that the protocol parser 64 can be instructed to perform a simple comparison operation, essentially matching strings, or it can perform more complex logical or arithmetic operations, such as bit shifting, ANDs, adds, length checking, and the like, depending on what may be required to identify an attack signature. In the present example, the GET request is an HTTP string amenable to a simple comparison, so no arithmetic or other types of operations are required. Other forms of attacks, however, may have signatures that require more complex processing to identify.

Once processing of the URI strings is complete, receipt of the "?" token causes the protocol parser 64 to shift states to the first parameter state, where it executes further instructions according to the "?" token. These instructions seek to identify those attacks that have the characteristic of simply possessing a Param1 field at this location. If any are identified, the appropriate SSID is sent to the attack detector 55.

Receipt of an "=" token shifts the protocol parser 64 to the next state, which examines the actual content of the Param1 field. Here, value1 would be parsed to determine whether it matches any of the strings in the tables 67. If so, the corresponding SSID is transmitted to the attack detector 55. Subsequent receipt of an "&" token shifts the protocol parser back to the parameter state, where this comparison is repeated for the Param2 field. The lack of any other tokens terminates the comparison process. Note that the HTTP token does not play a role in the processes of Figure 10 other than to indicate to the protocol parser 64 that HTTP is the appropriate protocol, indicating which tables of the tables 67 to use.

Figures 11A and 11B illustrate exemplary instructions contained within tokens according to an embodiment of the invention. Many of the instructions shown direct the protocol parser 64 to compare values within packet states to tabulated attack signatures. Note that this is not an exhaustive list and, while only comparison operations are shown, many other operations exist. More specifically, certain attacks leave signatures that can only be recognized upon mathematical, logical, or other manipulations on packet fields or states. Such operations, while not shown in Figures 11A and 11B, are nonetheless encompassed by the scope of the present invention. Thus, assuming complete implementation, the state machine of Figure 10 can detect attack signatures in any request method, detecting situations where a signature may be obscured by quoting and requesting further processing in that case. The same state machine can calculate the length of the entire request URI and compare it to a threshold. It can also calculate the length of each

IPR2024-00027
CrowdStrike Exhibit 1002 Page 728

URI query parameter name and value and compare it to a threshold. The state machine can detect attack signatures in any HTTP header field. It can also calculate the length of each HTTP header field and compare it to a threshold.

Figure 9 illustrates one hardware configuration in which the various components are located on one or more processors and information flows among them as shown by the arrows. This is, however, but one possible configuration. An alternate configuration is shown in Figure 12, where a SiByte processor acts as a common bus and processor, carrying out operations as commanded by the other components and moving information among them as shown. One of skill in the art can see that many other configurations are both possible and encompassed by the present invention.

The protocol parser 64 has now been fully described. Accordingly, attention now returns to Figure 2 and the remaining executable modules stored in memory 50. The classification and pattern-matching module 68 coordinates the comparison between the profiles of incoming and known intrusion signatures 70. As previously discussed, this module receives data from the fixed-field detector 63 and the protocol parser 64 and operates as an attack detector. In particular, the module receives Sub-Signature Identifications (SSIDs)
and determines if all the SSIDs required to form a complete pattern of a signature exist. The technique for distributing and updating the intrusion signatures 70 is discussed below.

Figure 2 also illustrates an encrypted session monitoring module 72 stored in memory 50. The encrypted session monitoring module 72 decrypts communications between a client computer and a server computer in accordance with public and private key information. In one embodiment, the decrypted information is clear text that can then be inspected for intrusions.

The encrypted session monitoring module 72 facilitates the complete monitoring of Secured Socket Layer (SSL) sessions so that it is possible to detect attacks launched through encrypted channels, such as HTTPS connections. The module 72 facilitates a non-intrusive method in that it is not visible to the normal SSL client and server applications and that a failure in the monitor will not interfere with normal SSL operation. The method also provides a means to avoid introducing any new security exposure to the SSL server.

The encrypted session monitoring module 72 addresses the problem of the increasing number of attacks launched through HTTPS connections. These attacks are not visible to current network based IDS solutions because the data is encrypted. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 729

monitoring module 72 allows a network intrusion detection system to see attacking packets. In the absence of this feature, one must install host based intrusion detection systems on each server.

To decrypt the SSL traffic, the sensor 22 uses the web server's public/private key information. If the sensor 22 is used to protect several web servers, it needs public/private keys from each server. Since the sensor 22 saves public/private keys from several servers, it stores the keys in a tamper proof storage, illustrated as protected key information 80.

The sensor monitors the key exchange protocol between the client and the server to derive the symmetric encryption key, the encryption initialization vectors (IV) and the message authentication code (MAC) key. The sensor can monitor the key exchange protocol because it has the server's public and private key. Once the key exchange is done, the sensor uses the symmetric encryption key and the encryption IV associated with the traffic flow to perform the decryption. Once decrypted, it is able to inspect the traffic for intrusions.

The sensor 22 preferably has a secure channel to the sensor management system 26 through which the sensor obtains the server's public and private keys. The sensor management system 26 does not store the public/private keys, although it provides an interface through which the key information can be supplied.

The protected key information 80 may be attacked in several manners. One form of attack is through the programmatic access of the keys by a normal administrator. A second form of attack is the physical security of the key storage. In the event that someone either obtains access to the unit or physically takes the unit, it should not be possible to read the keys, either programmatically or by probing of the electrical interface.

The encrypted session monitoring module 72 operations may be more fully characterized as follows. The sensor management system 26 sends the public/private key information of the server along with the server's Internet protocol address. The information is encrypted with the shared key of the sensor by the sensor management system 26. The sensor 22 decrypts the information and passes the public/private key to identify a unique KEY identifier. Every time the sensor sees key management requests for a particular server, it will send the session key exchange packets along with the KEY identifier to the encrypted session monitoring module 72. The module 72 derives the symmetric session encryption key, the encryption IV and the MAC key. Any encrypted traffic seen by the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 730

sensor is processed along with the KEY identifier. The module 72 decrypts the traffic using the symmetric encryption key, the encryption Ivs and returns the decrypted traffic.

The sensor 22 also includes a fail-over switch module 78. The fail-over switch module 78 supervises the operations performed by a sensor 22. In the event of the failure of that sensor 22, it switches control to a redundant sensor 24.

In order for the redundant sensor 24 to be in a position to operate, it must be synchronized with the primary sensor 22. The fail-over switch module 78 synchronizes the information contained on the primary sensor 22 and the redundant sensor 24.

The fail-over switch module 78 can operate in three configurations. One configuration is an active-standby state without load sharing. In this scenario, one sensor (e.g., sensor 22) is deployed as the active sensor and the other is deployed as the standby sensor (e.g., redundant sensor 26). The active unit passes state information to the standby unit every time it changes. After the fail-over, the new active unit has the same connection information as the failed unit. Preferably, the active and standby sensors are connected with a dedicated fail-over link 25, which is used to exchange state information. In one embodiment of the invention, the state information passed to the standby unit includes configuration information, TCP state information, UDP application information, SSL key data and other information necessary for the standby unit to take over processing if the active unit fails. If a standby unit powers up while the active unit is already operational, the standby unit will copy all the state information from the active unit. While the standby unit is copying the state information from the active unit, any updates to the copied state information are synchronized. When the active unit fails, the standby unit obtains the IP and MAC addresses of the active unit and starts accepting and forwarding traffic. When the active sensor and the standby sensor power up at the same time, the two sensor use a handshake protocol to figure out which will be the active unit and which will be the standby unit.

Another mode of operation is active/standby with load balancing. In this mode, both sensors receive traffic, although the primary link will have most of the network traffic. The secondary link will typically have control traffic. Each sensor will copy its per flow state information to the other sensor so any sensor can take over the responsibility of the other sensor when there is a fail-over. The two sensors exchange state information in real time. After the fail-over, the working sensor assumes the responsibility for the flows that were

IPR2024-00027
CrowdStrike Exhibit 1002 Page 731

handled by the failed sensor. In addition to providing redundancy, this model facilitates load balancing so both sensors can be utilized to provide advanced intrusion capabilities.

Since traffic can come to both sensors, the sensors use a load-balancing algorithm to determine which one processes the packets that belong to a particular flow. A flow is identified by the 5-tuple information that exists in each packet. In order to provide state-current intrusion detection, it is required that the packets directed to a flow are always processed where the flow was originally created. The load-balancing algorithm uses a hashing mechanism to determine which sensor will process the packet. Irrespective of which sensor sees the packets, based on the hash, the packets are assigned to one of the sensors.

A third redundant sensor configuration operates in an active/active with load balancing mode. In this mode, the two sensors are deployed to perform intrusion detection on four links. The traffic for a particular flow can come on any of the four links. Instead of monitoring one full duplex link, this configuration monitors four full duplex links. The two sensors are connected by four full links so each sensor can move packets to the other sensor. Each sensor has two line cards. The first line cards in each sensor are configured as a first pair and the second line cards in each sensor are configured as a second pair. To provide redundancy in this configuration, if a card in the base sensor 22 or if the entire sensor 22 goes down, the redundant sensor 24 is invoked.

When any line card in either sensor receives packets, the line cards use a hash algorithm to send packets to one of the line cards to process the packet. For a particular flow, irrespective of which line card receives a packet, the packets will always be sent to the same line card.

To implement redundancy, the first line cards in each sensor exchange state information. Similarly, the second line cards in each sensor exchange state information. In addition to maintaining its own information, each sensor maintains the state information for its counterpart line card in the other sensor. If the first line card in the first sensor goes down, all traffic meant for the first line card is delivered to the first line card in the redundant sensor. Similarly, if the second line card in the first sensor goes down, all traffic meant for the second line card is diverted to the second line card in the redundant sensor. If the base sensor 22 goes down, the traffic from the first line card and second line card in the base sensor is directed to the corresponding line cards in the redundant sensor.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 732

Thus, the fail-over switch module 78 coordinates state redundancy for intrusion detection systems. In addition, it provides load balancing.

The features of an embodiment of the sensor 22 have now been fully described. Attention now turns to a description of the sensor management system 26. Figure 13 illustrates an embodiment of a sensor management system 26. This implementation is in the form of a general-purpose computer, including a central processing unit 100 and a set of input/output devices 102, which communicate with one another over a system bus 104. The input/output devices 102 may include a keyboard, mouse, video monitor, printer, and the like.

Also connected to the system bus 104 is a memory 106, which may be primary and/or secondary memory. The memory 106 stores a set of executable programs utilized to implement features of the invention. In particular, the memory 106 stores a sensor control module 108, which performs general supervisory operations over one or more sensors 22

The sensor management system 26 also includes a virtual intrusion detection system (VIDS) provisioning module 110. This module performs local virtual VIDS operations as coordinated by the corresponding VIDS provisioning module 130 associated with the global sensor management system 34. The operation of VIDS provisioning module 110 is discussed below in connection with the VIDS provisioning module 130 of the global sensor management system 34.

The sensor management system 26 also stores a real-time signature update module 112. This module coordinates the delivery of intrusion signatures 70 to sensors 22. This module is responsive to control signals from the update server 38, as discussed below.

Figure 14 illustrates an embodiment of a global sensor management system 34. This implementation is in the form of a general-purpose computer, including a central processing unit 120 and a set of input/output devices 122, which communicate with one another over a system bus 124. The input/output devices 122 may include a keyboard, mouse, video monitor, printer, and the like.

Also connected to the system bus 124 is a memory 126, which may be primary and/or secondary memory. The memory 126 stores a set of executable programs utilized to implement features of the invention. In particular, the memory 126 stores a system management module 128, which performs general supervisory operations. The memory also stores a VIDS provisioning module 130.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 733

The VIDS provisioning module 130 operating with the provisioning module 110 of the sensor management system 26 enables service providers (e.g., Internet Service Providers or Managed Security Service Providers) to supply intrusion protection to multiple subscribers with one integrated platform. More particularly, the VIDS provisioning module 130 allows different subscribers to view designated traffic information associated with their logical traffic paths, as defined in the sensors.

The VIDS provisioning module 130 facilitates protection for multiple traffic streams for different subscribers using a single sensor platform while maintaining security separation. The module 130 supports different security policies, protection objectives, and security management preferences effectively and efficiently. The module 130 reduces the management task and total cost of providing intrusion detection security services.

The VIDS module 130 allows a user to specify a subscriber's view and a subscriber configuration. A VIDS view restricts the intrusion information shared with a subscriber to the subset related to the subscriber configuration. A subscriber is allowed to use VIDS interfaces to configure the VIDS as if it were a physical device.

The VIDS provisioning module 130 operates in conjunction with a sensor 22 and the update server 38. As discussed below, the update server 38 is used to notify customers of new software or network intrusion signature updates. The update server 38 also provides the software images and signature files for download. As previously discussed, the sensor 22 performs signature matching against network traffic and generates responses in case of intrusions. A sensor 22 accepts configuration and control messages and sends intrusion alerts and/or events to the sensor management system 26. The requirement for a sensor to support VIDS is simple. Since each VIDS can customize the response in the face of an intrusion, a sensor must be able to generate VIDS specific responses. In addition, the sensor 22 labels all alerts and events sent to the sensor management system 34 with VIDS identifiers so that the sensor management system 34 does not have to spend extra computer resources in filtering VIDS events.

The VIDS provisioning module 130 operating on the sensor management system 34 is central to the VIDS implementation. It stores VIDS definitions and enforces VIDS view, configuration, and access control restrictions using a role-based system. The VIDS module 130 provides a window into the sensor management system 34 for the IDS service provider's customer, also referred to as the subscriber. The service provider has made available a part of the sensor for subscriber use and the virtual IDS provides the subscriber

IPR2024-00027
CrowdStrike Exhibit 1002 Page 734

with the capability to manage the allocated resources. Virtual IDS resource allocation is based on one of the following: (1) sensor identification and port number; (2) VLAN ID for traffic belonging to a subscriber; and (3) IP address range that belongs to the subscriber. There are several major IDS components involved in supporting the VIDS features: user and security management, user interface, intrusion signature and response configuration, and real-time VIDS recognition in the sensor. These components are discussed below.

The VIDS provisioning module 130 implements a role based management system. A user role is defined as a group of actions that can be performed on a given device. It is the basis of the authorization framework. Roles must be tied to a resource.

There are two kinds of resources in the sensor management system: the sensor management system itself and the VIDS. Anyone with a sensor management system role is automatically mapped to all system-wide operations, since the sensor management systems 34 in the network 20 mirror one another. Each VIDS has a unique identifier (ID). The VIDS provisioning module 130 enforces security.

In one embodiment of the invention, there are five role types: system administrator, analysis/operator, user management, intrusion expert, and monitoring. A system administrator is allowed to perform simple administrative tasks, such as start/stop processes and database configuration. An analysis/operator is allowed to perform routine maintenance tasks. A user management role allows certain management tasks, such as adding or deleting role assignments. An intrusion expert is a person responsible for configuration and monitoring of intrusion related matters. A monitoring role provides read-only access.

Preferably, the VIDS provisioning module 130 implements a graphical user interface (GUI). The GUI enables the subscriber to access the sensor management system 34 to manage and monitor the VIDS. The GUI can be implemented within a browser, either as an applet or using HTML.

Those skilled in the art will recognize a number of advantages associated with the VIDS provisioning module 130. First, it supports flexible delegation of administrative tasks in a scalable manner. In a large-scale service provider environment, this is of particular importance considering the large number of potential intrusion events that could easily overwhelm a centrally managed system. Second, the VIDS opens a window for subscribers to view information related to their own systems that is otherwise unavailable. Third, it

IPR2024-00027
CrowdStrike Exhibit 1002 Page 735

reduces the total cost of ownership for the service provider and its subscribers. Fourth, it provides an easy to deploy and manage solution for a large number of subscribers.

The global sensor management system 34 of Figure 14 also stores a DDOS tracking monitor 132. The concept of a distributed denial of service (DDOS) attack has been previously introduced and discussed in connection with the sensor 22. The DDOS tracking monitor 132 includes executable code to implement multi-sensor collaborative techniques in identifying DDOS attacks. The operation of the DDOS tracking monitor 132 is more fully appreciated in connection with Figure 1. Sensor 27_0 monitors traffic on the physical link connecting the protected server 32 to the enterprise network 30. Sensors 27_1 through 27_N monitor the links that connect the enterprise network 30 to Internet service providers 28_1 through 28_N. Although the sensors 24 and 26 may be deployed to monitor links on the outside of firewalls, on the inside of firewalls, or both, this fact is not distinguished in this example. As previously indicated, the sensors 22 are managed by a hierarchy of sensor management systems 26 and the global sensor management system 34.

The DDOS tracking monitor 132 implements multi-sensor collaboration as follows. All sensors 22 are controlled by the global sensor management system 34. The sensors may be deployed initially by the management system 34 or, when activated, they can register with the management system 34. The sensors are identified by their topological location with respect to the target of protection (i.e., protected server 32). Sensor 27_0 sees the aggregate of all traffic passing through sensors 27_1 through 27_N that is destined for the protected server 32. The global management system 34 pushes statistical profiles to all of the upstream sensors 27_1 through 27_N.

Each sensor 22 or 24 inserts probe packets or path markers in the traffic it processes. The DDOS tracking monitor 132 specifies the frequency for generating these markers. The frequency is dependent upon the characteristics of the target, such as the information it stores and the vulnerability to attack. The frequency is also dependent upon the relative distance between the sensor and the target.

In one embodiment, the DDOS tracking monitor 132 correlates sensor information to detect a DDOS attack (in other embodiments, DDOS detection can be performed remotely and the results transmitted to the tracking monitor 132). The correlation rules utilized by the DDOS tracking monitor to detect an attack may consider any number of factors. For example, a rate increase of greater than X Mbps/sec observed by Y active sensors comprising greater than Z% of the total contributing links characterizes a pending

IPR2024-00027
CrowdStrike Exhibit 1002 Page 736

DDOS attack on the target coming from the links crossing the threshold. In this example, the selection of X, Y, and Z is based on the tolerable load at the target as well as the latency to respond to the threat (e.g., activation of rate limiting at the corresponding boundaries). Another example correlation rule tracks every active sensor and generate an alert when traffic exceeds a predetermined high water mark for a given zone, such as a single server, a sub-net, an enterprise network, and the like. As yet another example, a novel attack detection method is also disclosed in the co-pending U.S. patent application referenced previously. These and other methods of DDOS attack detection are encompassed by the present invention, which in this aspect simply discloses the detection of DDOS attacks.

Thus, the DDOS tracking monitor 132 operates in conjunction with a set of provisioned sensors that insert path markers into network traffic directed toward a protected computer. The DDOS tracking monitor 132 identifies when the path markers and network traffic match a distributed denial of service attack profile. In response to this situation, the DDOS tracking monitor 132 initiates measures to deny access to the protected computer. The DDOS tracking monitor 132 may subsequently be used to analyze the path markers to identify a source of a distributed denial of service attack.

The components of the sensor management system 34 have now been fully described. Therefore, attention turns to a description of the update server 38, which is depicted in Figure 15. The update sever 38 of Figure 15 includes standard computer components of a central processing unit 140 and a set of input/output devices 142, which communicate over a system bus 144. A memory 146 is also connected to the system bus 144. The memory stores a set of executable programs to implement the functions of the update server. In particular, the memory stores a secure update download module 148 and a hierarchical attack categorization module 150.

The secure update download module 148 includes executable code to download an attack file 149 to a sensor management system 26. The attack file 149 specifies attacks and counter measures. Preferably, the file also identifies unknown attacks and suggests responses for such attacks. The attack file 149 includes information forming the intrusion signatures 70 processed by the classification and pattern-matching module 68.

The attack file 149 can be downloaded using different approaches. The secure update download module 148 can periodically download the attack file 149. Alternately, the secure update download module can download the attack file 149 in response to a request from a sensor management system 26. Alternately, email alerts may also be used to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 737

deliver updated attack files 149. The SMS alert notification module 152 may be used to send alerts by secure e-mail (e.g., SMIME) when a new signature update is available for download.

In one embodiment of the invention, the sensor management system 26 uses HTTP/SSL to communicate with the secure update download module 148. Since the sensor management system 26 resides behind the firewall 36, it is easier to utilize a PULL methodology to download the rules. A PUSH methodology requires opening holes in the firewall. The sensor management system 26 authenticates itself to the update server 38 prior to receiving an attack file 149.

The attack file 149 may be formed utilizing the hierarchical attack categorization module 150 of the invention. The hierarchical attack categorization module 150 is used to develop, deploy and manage information on different computer attacks and countermeasures. In one embodiment of the invention, potential attack space is defined in terms of a set of categories, for example, including probes and port scans, compromise of confidentiality, compromise of authentication, compromise of data integrity, denial of services, and direct violation of security policies. Dividing the potential attack space in this manner allows one to have greater insight into the nature of the attack and its manifestations. Accordingly, it is easier to tailor a response to the attack.

Probes and port scan attacks represent reconnaissance efforts from an attacker in preparation for additional attacks. Examples of this category include pings for live hosts, tests for operating system types, and TCP port scans for active Internet services. Compromise of confidentiality attacks cause direct expose to sensitive information, such as certain system files, director structures, or other user files. Examples in this category include sniffing of a sub-net, transmitting confidential information via stealthy (e.g., HTTP tunneled) channels, or forcing system core dumps.

Compromise of authentication/authorization attacks entail an attacker obtaining unauthorized access privileges. There are often more serious consequences from such attacks if they are not dealt with quickly. For example, that attacker can install backdoors to allow easy access for future attacks. Examples of this category include the well-known send mail attack to obtain illegal shells on a local machine, TCP connection hijacking, and the set of buffer-overflow related attacks.

A compromise of data integrity entails the corruption of sensitive data. It can also have secondary impact if not detected or rectified quickly. For example, a successful

IPR2024-00027
CrowdStrike Exhibit 1002 Page 738

"binmail" attack can result in an altered ".rhosts" file, which gives the attacker future logins without a password.

A direct violation of security policy is another form of attack. Although this category most often produces an impact characterized by one of the previous five categories, sometimes the misuse may not produce any observable impact, or at least not immediately. If one recognizes this special category and recognizes that there are well-defined policies governing the use of privileges, one can detect the misuse as soon at it occurs, even in the absence of a serious impact.

A final form of attack associated with this example is the denial of service attack. This type of attack is discussed above.

In addition to defining attack categories, it is useful to define attack target platforms. In one embodiment of the invention, the following hierarchy is identified: (1) vendor (e.g., SUN, IBM, HP, SGI, CISCO), (2) device (e.g., workstation, server, router, firewall), (3) operating system (e.g., Solaris, Linux, MAC OS), and (4)version/revisions.

Attack target services can also be categorized. By way of example, target services may include: HTTP, DNS, SMTP, FTP, telnet, NFS, and the like. Attack launch platforms may also be categorized. The same hierarchical structure can be used as in the case of an attack target. Monitoring points may also be defined. Monitoring points refer to the places or platforms from which any event related to an attack can be observed. The same general hierarchy for describing the attack target can be used for describing the monitoring points, except that for attacks launched across the network (as opposed to a local user from the console) any link on the path should be considered a viable monitoring point.

Each identified attack is assigned a serial number, called a sigID. Anomaly-based detection mechanisms can also be assigned a unique sigID so that they can be assigned to a coverage map. The coverage map can be viewed as an attack categorization tree with detection sigIDs attached to the nodes of the tree, as shown in Figure 16. The tree is created using categorization dimensions of the type discussed above. At the top level all attacks are mapped according to the attack impact. At the next level, the attacks are further categorized based on the attack target platform, which is itself a hierarchy. Any attack is only attached to one node of the tree. For example, attack a1 is attached to the denial of service node because it is a general denial of service attack that applies to all the platforms. Attack a3 is only attached to the leaf node v2.6 because it only affects SUN Solaris 2.6

IPR2024-00027
CrowdStrike Exhibit 1002 Page 739

Detected signature identifications (with anomaly detection mechanisms treated as special signatures) are also attached to the tree node. Again, a more general signature will be attached to nodes closer to the root because they cover larger numbers of specific target platforms, while the most specific signature is attached to a leaf node, such as sig5.

Figure 17 illustrates processing steps performed by an embodiment of the hierarchical attack categorization module 150 of the invention. The module initially constructs a hierarchy characterizing different computer attacks and countermeasures (block 160). This operation results in a structure of the type shown in Figure 16. The hierarchy is then traversed to identify computer attacks and countermeasures relevant to the target platform (block 162). Detection and protection measures for the target platform are then collected (block 164). This can result in an attack file 149. A sensor is then supplied, through a download, with the protective software (e.g., the attack file) for the target platform (block 166).

The hierarchical approach of the invention allows one to clearly describe, in one data structure, what attacks a given environment is vulnerable to, what protection means (mechanisms and signatures) are available, and how detection alerts are correlated. Processes for utilizing this representation constitute effective methods for deployment and operation of the solution, for analysis of alerts, and for security management.

It should be recognized that the various novel components of the architectures described above can be combined to afford users maximum protection against attacks. For instance, DDOS source detection, anomaly-based intrusion detection, and signature-based intrusion detection can be implemented on a redundant sensor architecture to make them more robust. Similarly, the update server 38 can be configured to update both sensors 22 and redundant sensors 24 with updated attack signatures, anomaly profiles, and the like. The various attack detection methods (i.e., anomaly-based, signature-based, etc.) can also be performed on information decrypted by the various decryption methods of the invention. These and other combinations should be apparent from the previous discussion.

The foregoing description, for purposes of explanation, used specific nomenclature to provide a through understanding of the invention. However, it will be apparent to one skilled in the art that specific details are not required in order to practice the invention. Thus, the foregoing descriptions of specific embodiments of the invention are presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed; obviously, many modifications and variations

IPR2024-00027
CrowdStrike Exhibit 1002 Page 740

are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, the thereby enable other skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the following claims and their equivalents define the scope of the invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 741

What is claimed is:

1. A method of detecting intrusions on a computer, comprising:

identifying an internet protocol field range describing fields within internet protocol packets received by a computer;

establishing a connectivity range describing a distribution of network traffic received by said computer;

determining an internet protocol field threshold and a connectivity threshold from said internet protocol field range and said connectivity range, respectively;

during the operation of said computer, calculating values for said internet protocol field range and said connectivity range; and

comparing said values to said internet protocol metric threshold and said connectivity metric threshold so as to identify an intrusion on said computer.

2. A method of detecting intrusions on a computer, comprising:

storing an intrusion signature describing an attack on a computer;

receiving a plurality of internet protocol packets, said plurality of internet protocol packets collectively containing an information sequence within a series of states;

rearranging said plurality of internet protocol packets so as to place said information sequence in order; and

successively examining each state of said series of states so as to correlate said information sequence to said intrusion signature.

3. A method of forming a virtual intrusion detection system, comprising:

positioning a set of sensors in a network environment, each sensor supporting multiple logical traffic paths; and

providing a set of sensor management systems corresponding to said set of sensors, said set of sensor management systems enforcing a set of virtual intrusion detection systems, wherein each virtual intrusion detection system corresponds to a predetermined logical traffic path through said set of sensors, each virtual detection system providing sensor traffic information solely to authorized parties.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 742

4. A method of provisioning a computer against computer attacks, comprising:

constructing a hierarchy characterizing different computer attacks and counter measures;

traversing said hierarchy to identify computer attacks and countermeasures relevant to a target platform;

collecting detection and protection measures in response to said traversing; and

downloading said detection and protection measures to a security sensor associated with said target platform.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 743

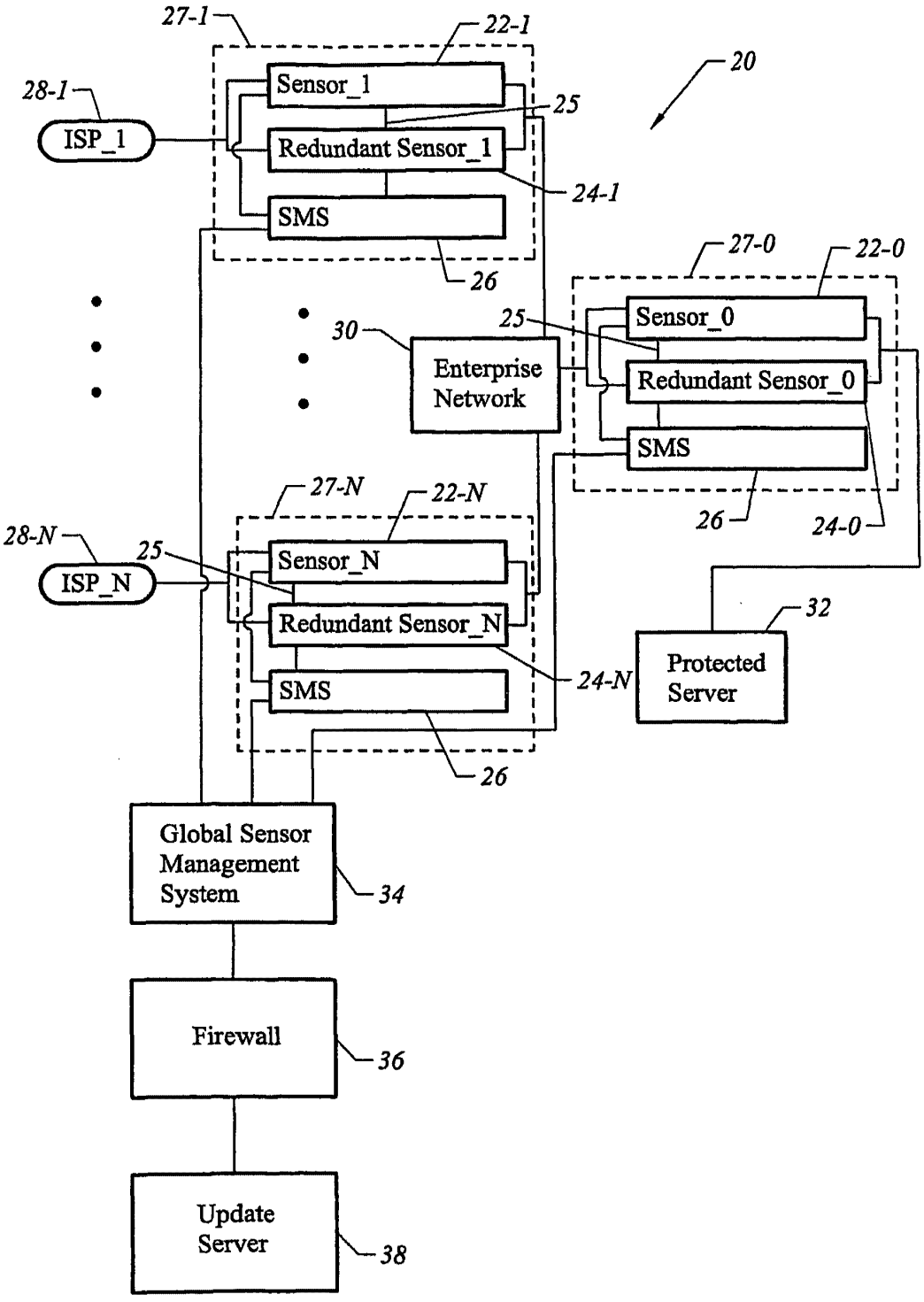


FIG. 1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 744

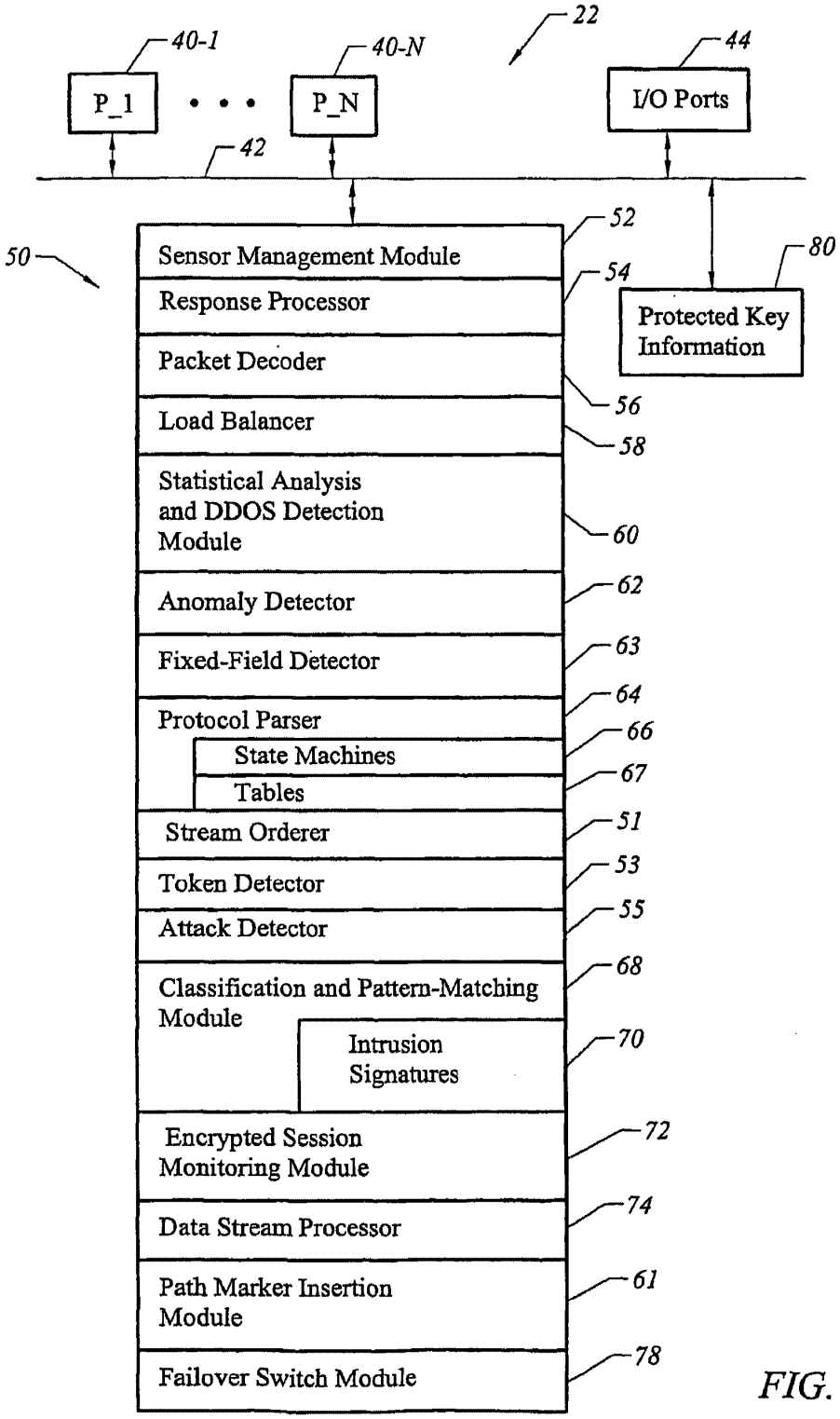


FIG. 2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 745

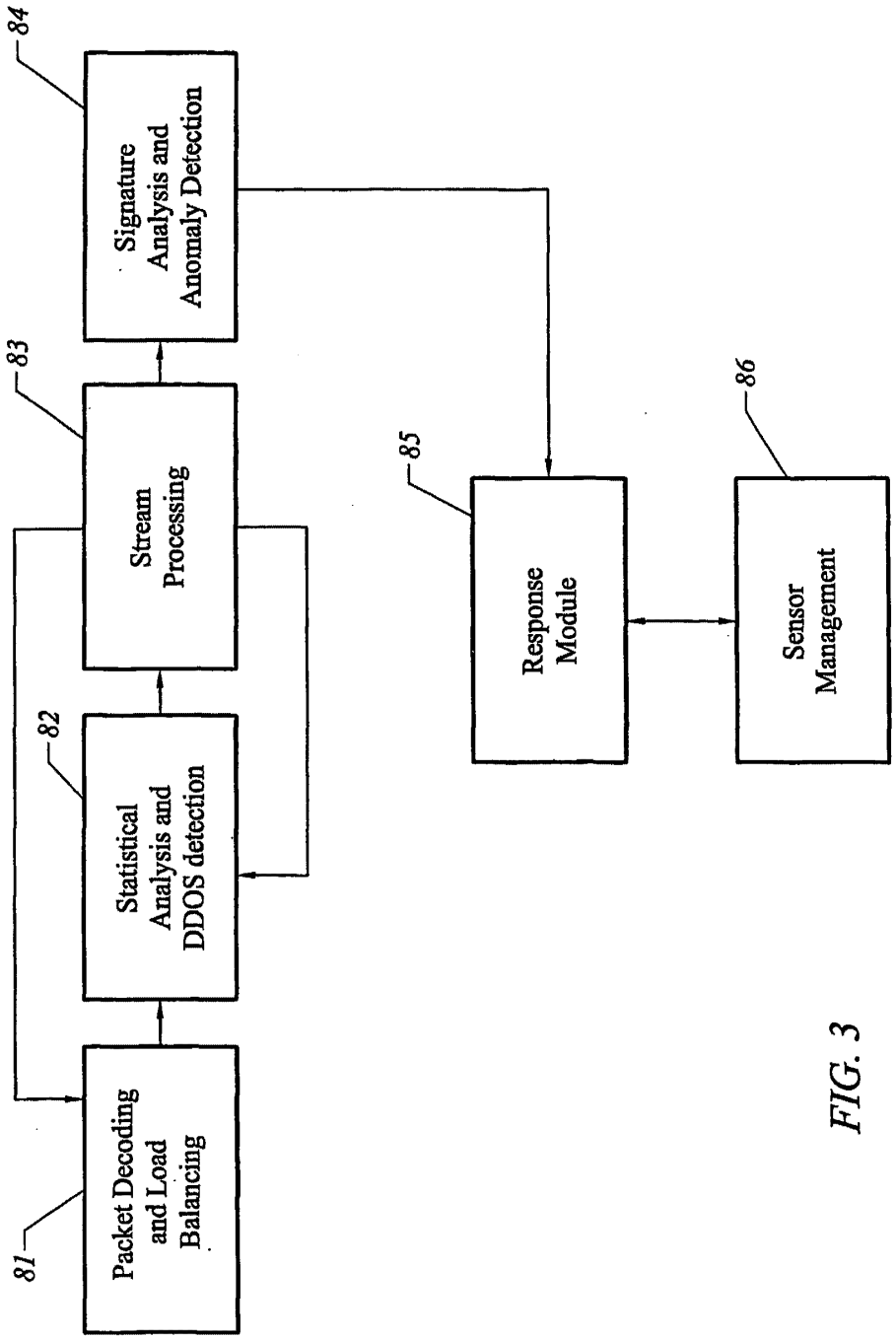


FIG. 3

IPR2024-00027
CrowdStrike Exhibit 1002 Page 746

SUBSTITUTE SHEET (RULE 26)

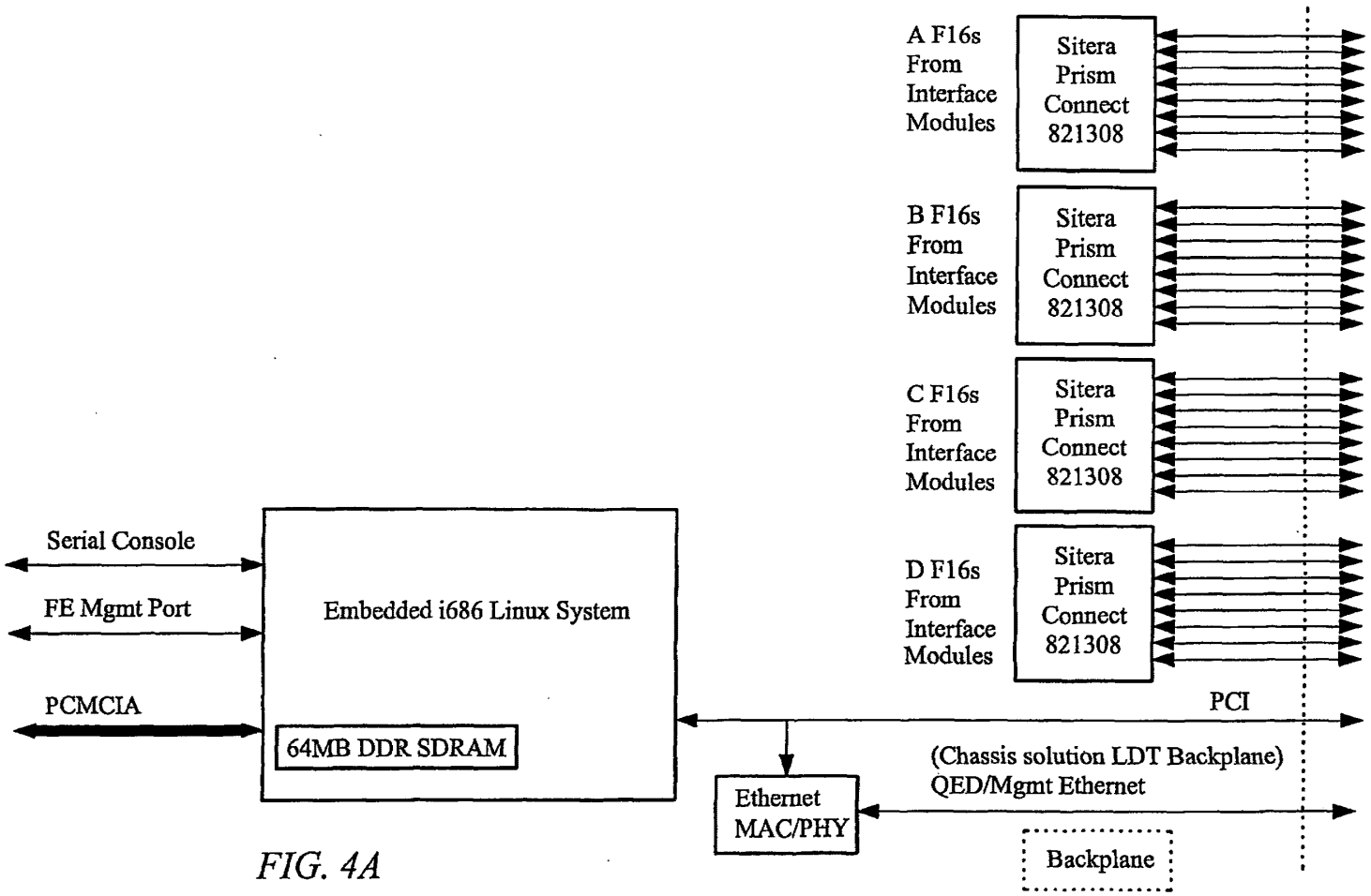


FIG. 4A

IPR2024-00027
CrowdStrike Exhibit 1002 Page 747

SUBSTITUTE SHEET (RULE 26)

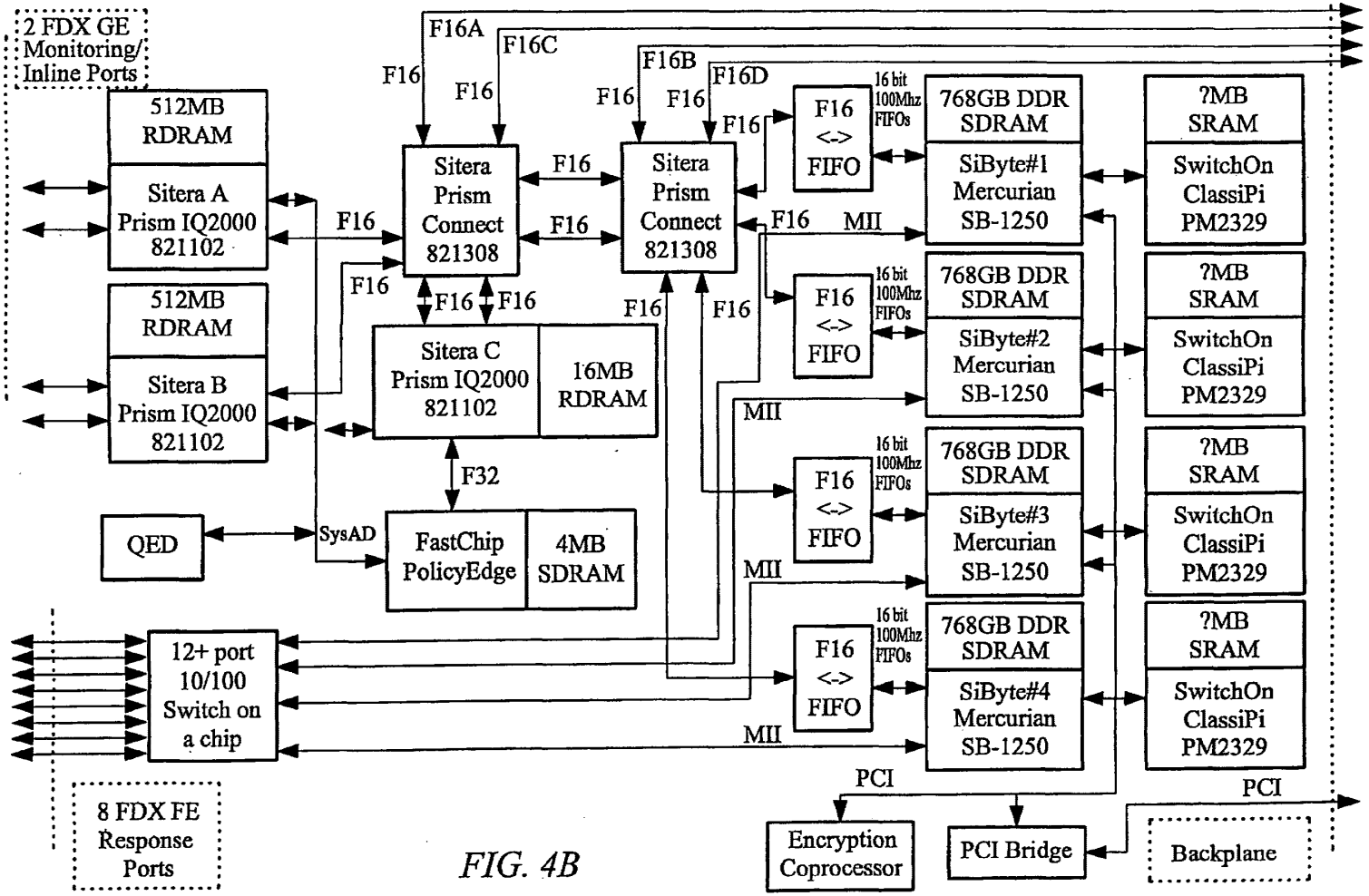


FIG. 4B

IPR2024-00027
CrowdStrike Exhibit 1002 Page 748

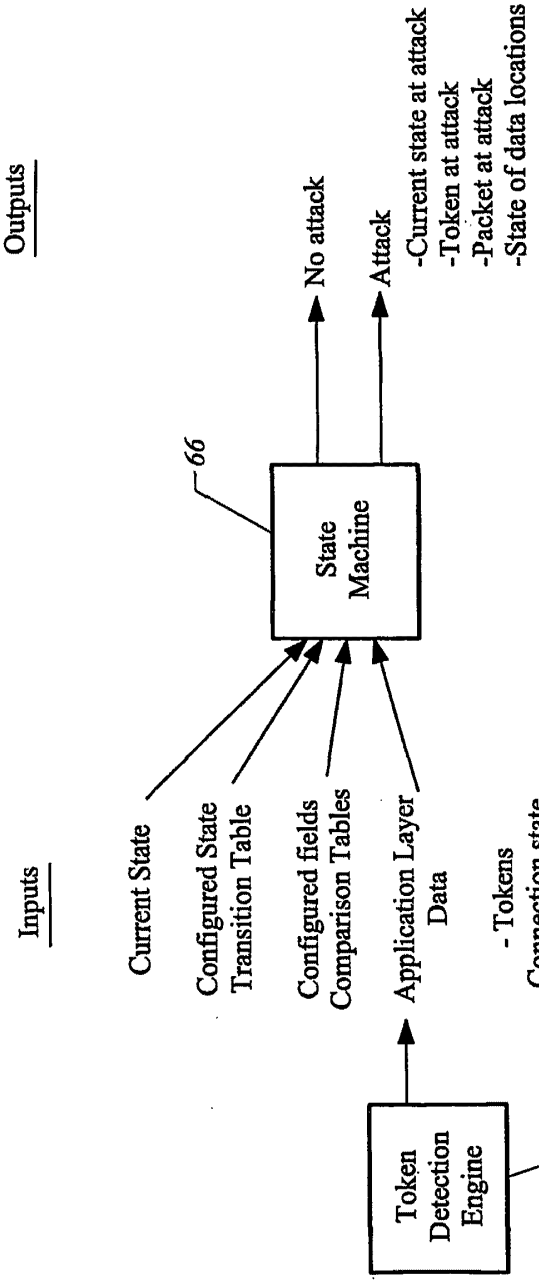


FIG. 5

IPR2024-00027
CrowdStrike Exhibit 1002 Page 749

SUBSTITUTE SHEET (RULE 26)

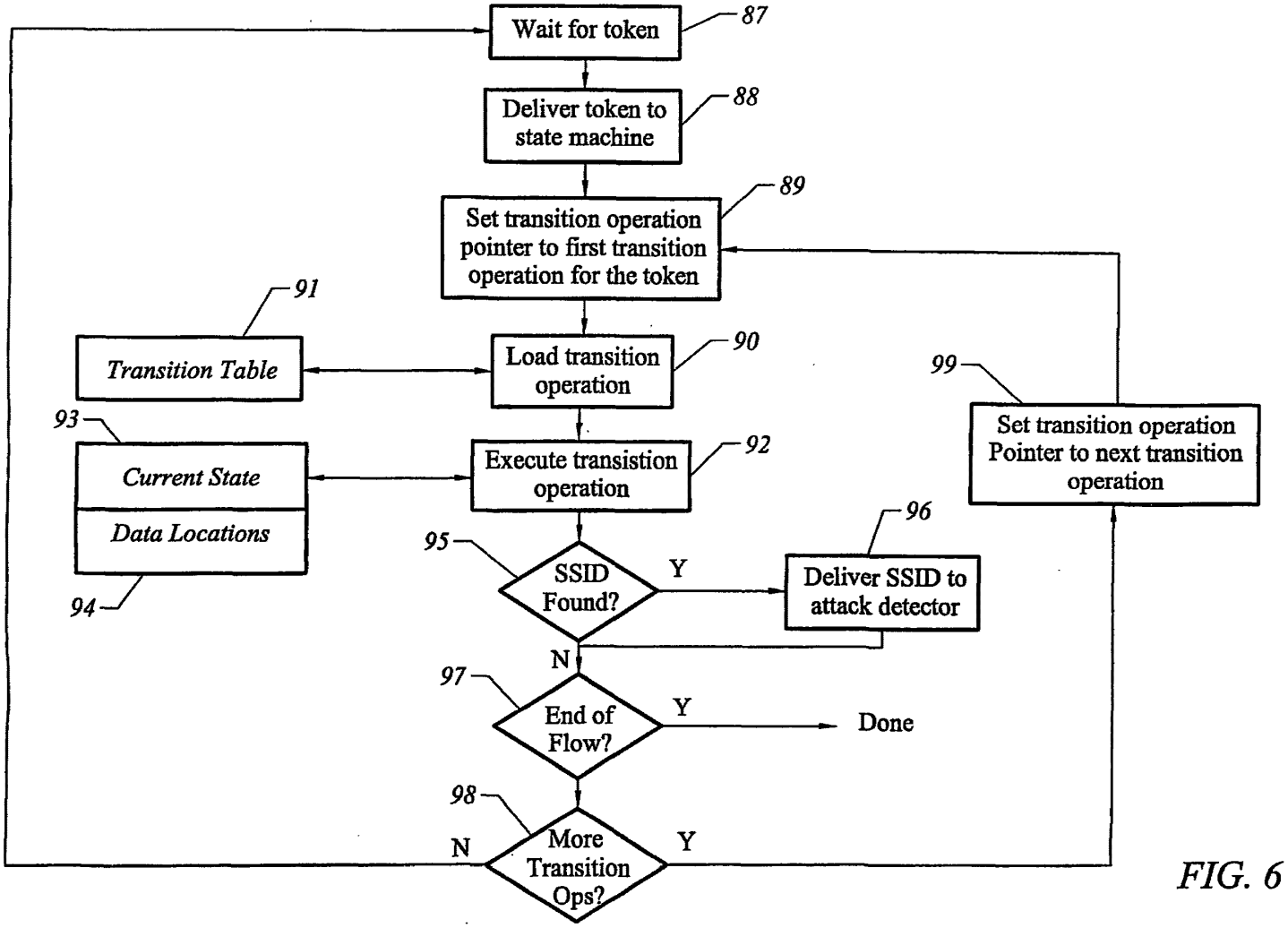


FIG. 6

IPR2024-00027
CrowdStrike Exhibit 1002 Page 750

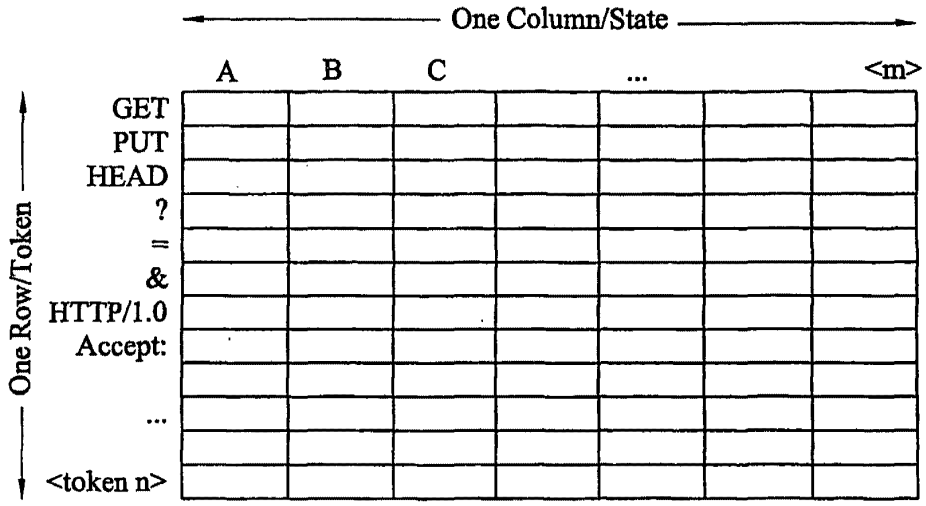


FIG. 7

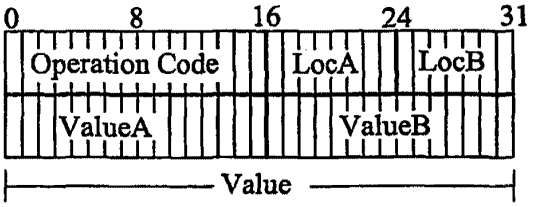


FIG. 8

IPR2024-00027
CrowdStrike Exhibit 1002 Page 751

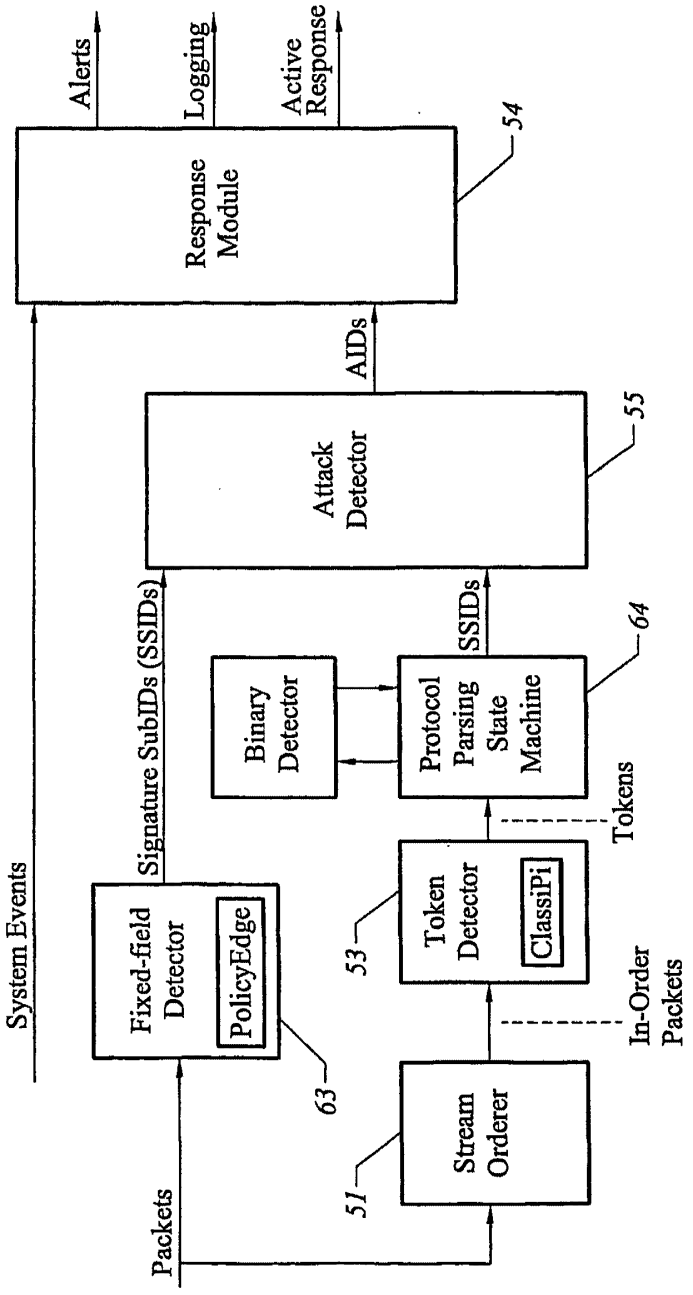


FIG. 9

IPR2024-00027
CrowdStrike Exhibit 1002 Page 752

SUBSTITUTE SHEET (RULE 26)

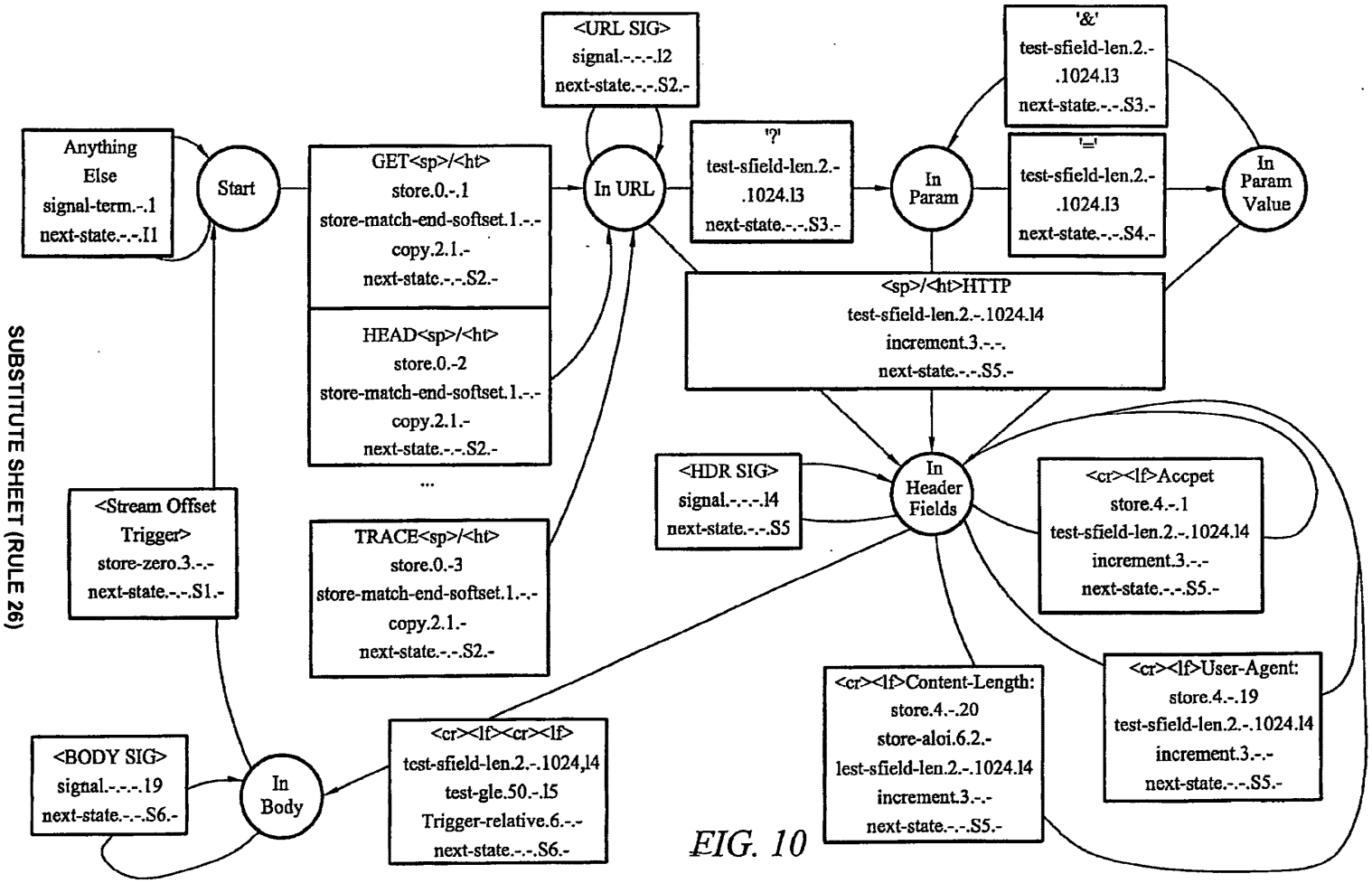


FIG. 10

iflt	if LocB≠0 compare LocA to LocB else compare LocA to Value field if LocA < Value execute following ops else skip following ops endif endif
iflte	if LocB ≠ 0 compare LocA to LocB else compare LocA to Value field if LocA ≤ Value execute follwoing ops until next else or fi else skip following ops endif endif
ifeq	if LocB ≠ 0 compare LocA to LocB else compare LocA to Value field if LocA = Value execute following ops until next else or fi else skip following ops endif endif
ifneq	if LocB ≠ 0 compare LocA to LocB else compare LocA to Value field if LocA = Value execute following ops until next else or fi else skip following ops endif endif
ifgte	if LocB ≠ 0 compare LocA to LocB else compare LocA to Value field if LocA ≥ Value

FIG. 11A

IPR2024-00027
CrowdStrike Exhibit 1002 Page 754

	execute following ops until next else or fi else skip following ops endif endif
ifgt	if LocB ≠ 0 compare LocA to LocB else compare LocA to Value field if LocA > Value execute following ops until next else or fi else skip following ops endif endif
ifql	if (locA = ValueA) and (LocB = ValueB) execute following ops until next else or fi else skip following ops endif
else	halt/resume instruction exection
fi	endif

FIG. 11B

IPR2024-00027
CrowdStrike Exhibit 1002 Page 755

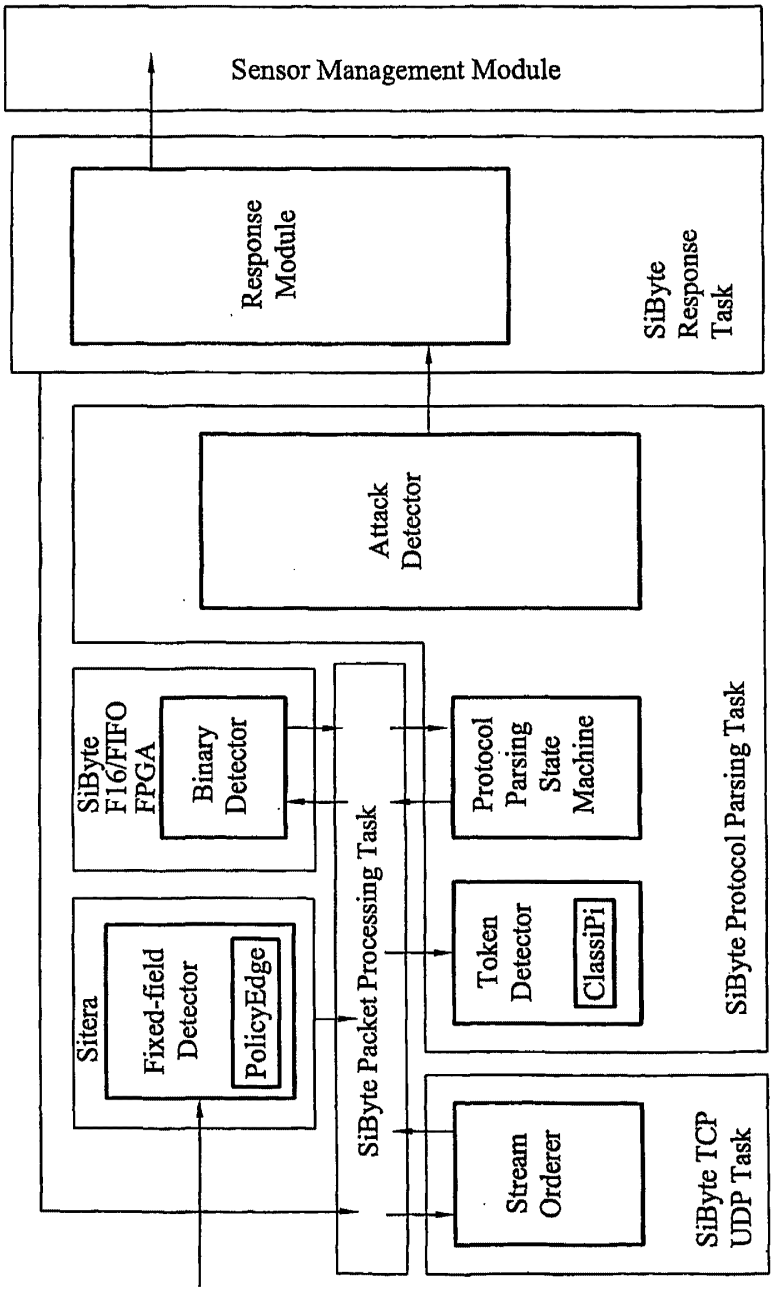


FIG. 12

IPR2024-00027
CrowdStrike Exhibit 1002 Page 756

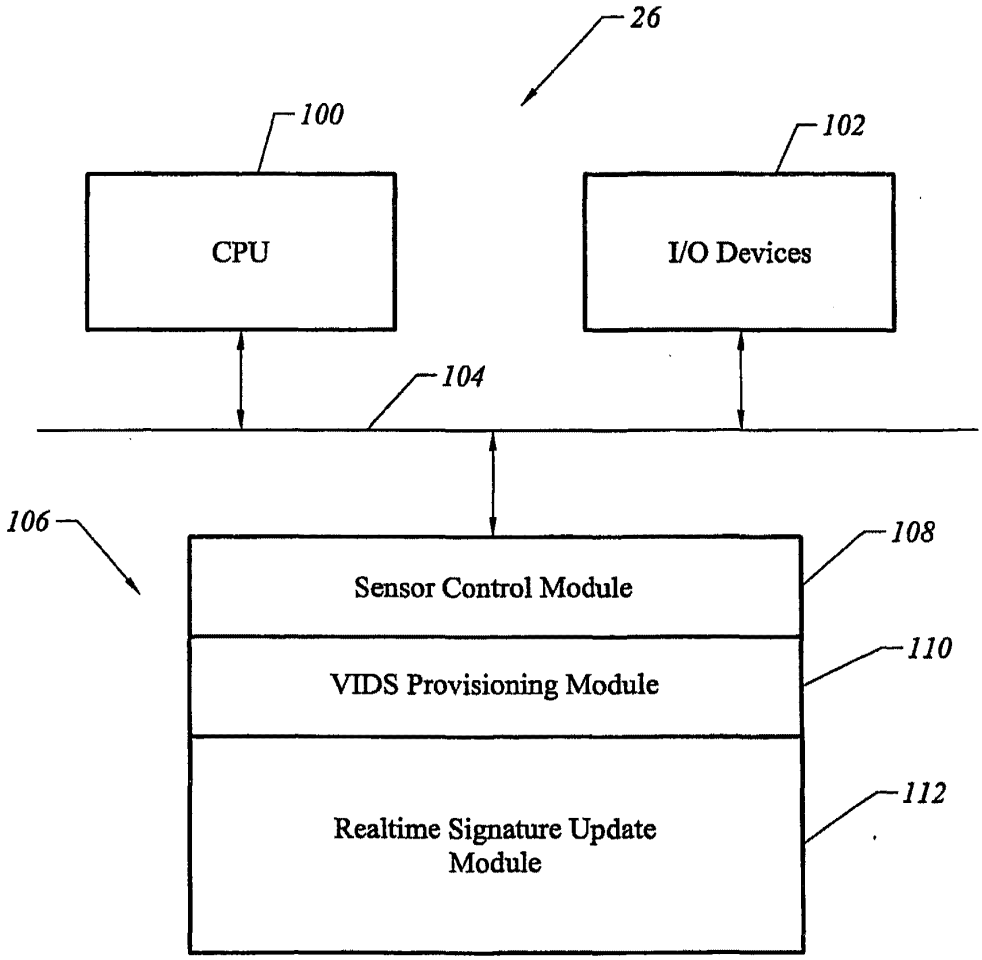


FIG. 13

IPR2024-00027
CrowdStrike Exhibit 1002 Page 757

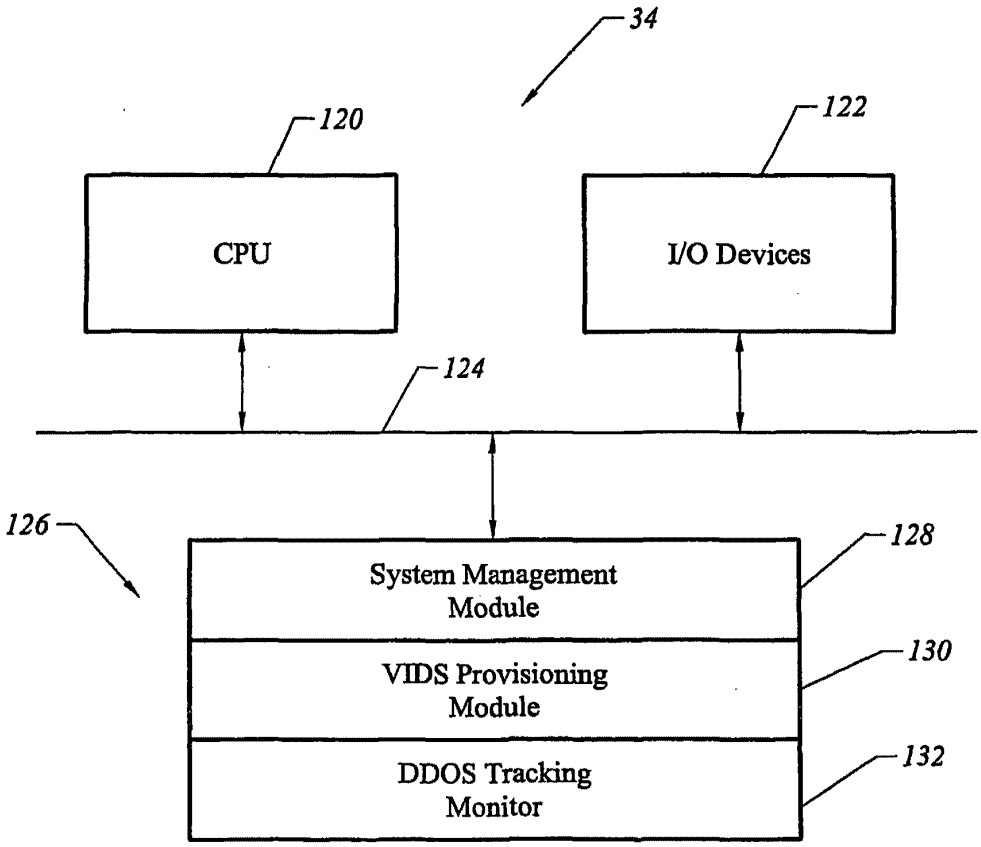


FIG. 14

IPR2024-00027
CrowdStrike Exhibit 1002 Page 758

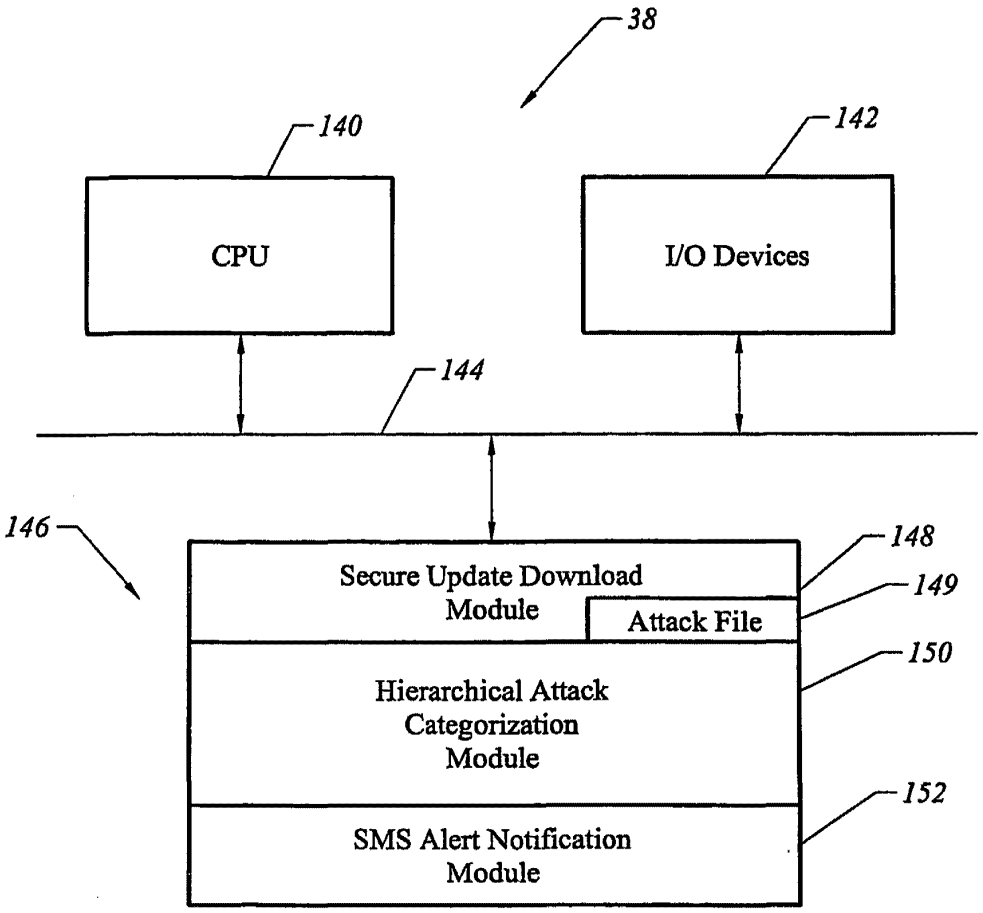


FIG. 15

IPR2024-00027
CrowdStrike Exhibit 1002 Page 759

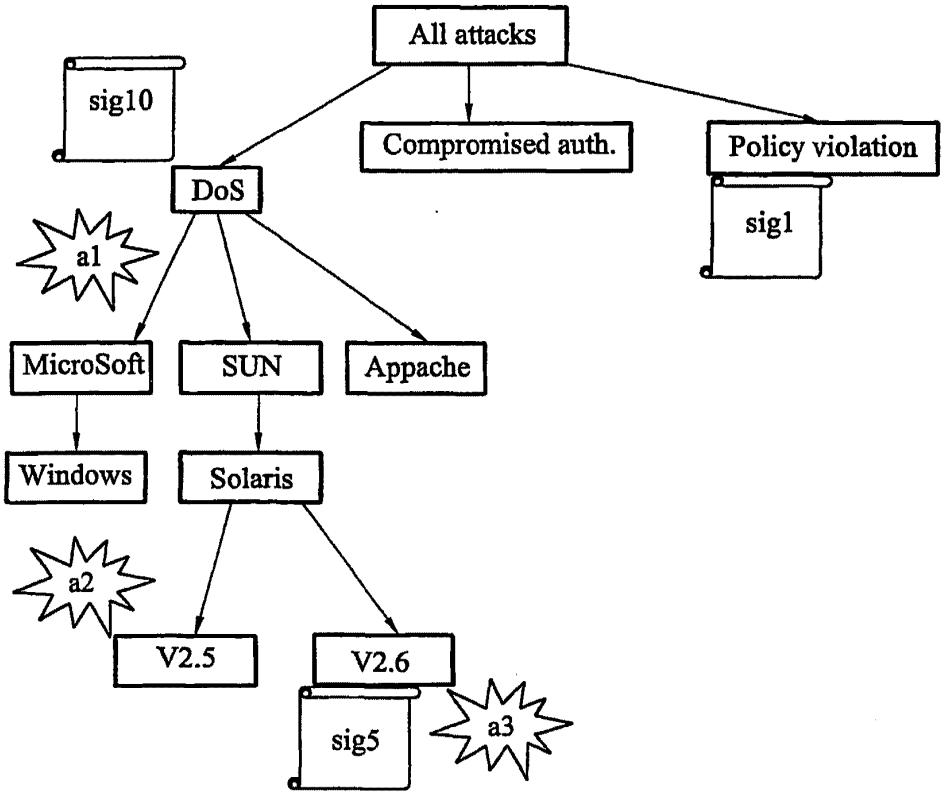


FIG. 16

IPR2024-00027
CrowdStrike Exhibit 1002 Page 760

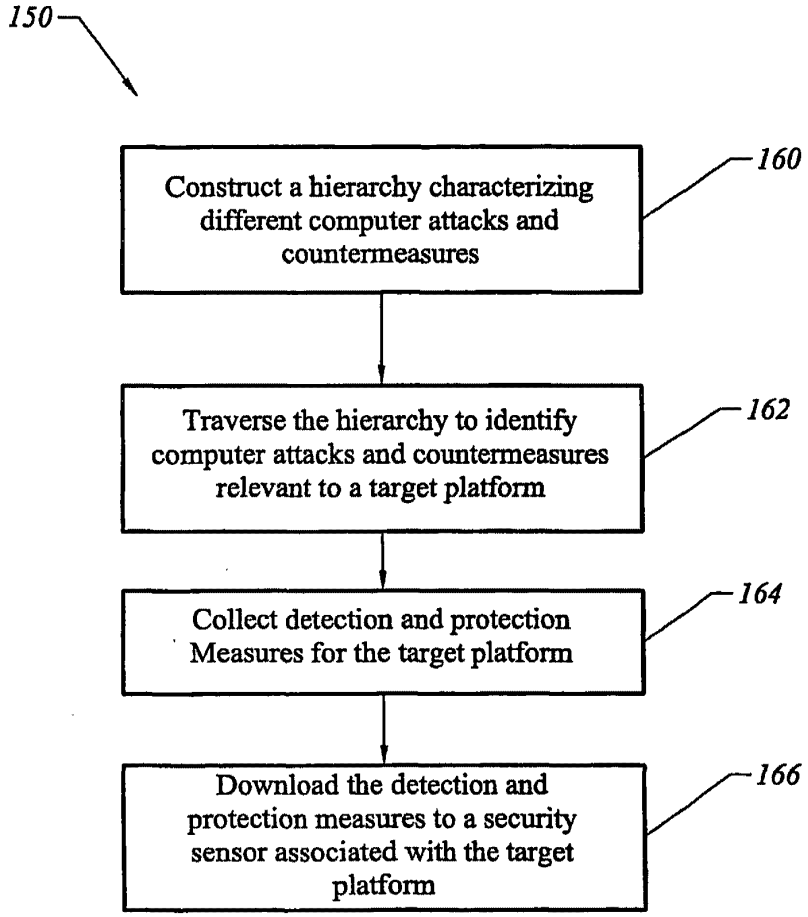


FIG. 17

IPR2024-00027
CrowdStrike Exhibit 1002 Page 761

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 985 995 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
13.08.2003 Bulletin 2003/33

(51) Int Cl.7: **G06F 1/00**

(21) Application number: **98117083.0**

(22) Date of filing: **09.09.1998**

(54) **Method and apparatus for intrusion detection in computers and computer networks**

Verfahren und Vorrichtung zur Eindringdetektion in Rechnern und Rechnernetzen

Procédé et appareil de détection d'intrusion dans des ordinateurs et des réseaux d'ordinateurs

(84) Designated Contracting States:
DE FR GB

(43) Date of publication of application:
15.03.2000 Bulletin 2000/11

(73) Proprietor: **International Business Machines Corporation**
Armonk, NY 10504 (US)

(72) Inventors:
- **DACIER, Marc C.**
  8134 Adliswil (CH)
- **DEBAR, Hervé C.**
  8134 Adliswil (CH)
- **WESPI, Andreas A.**
  8155 Nettmenhasli (CH)
- **FLORATOS, Aris**
  Long Island City, New York 11106 (US)
- **RIGOUTSOS, Isidore**
  Astoria, New York 11103 (US)

(74) Representative: **Williams, Julian David**
International Business Machines Corporation
Säumerstrasse 4/Postfach
8803 Rüschlikon (CH)

(56) References cited:
US-A- 5 621 889

- **DENAULT S ET AL: "INTRUSION DETECTION: APPROACH AND PERFORMANCE ISSUES OF THE SECURENET SYSTEM" COMPUTERS & SECURITY INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, vol. 13, no. 6, 1 January 1994, pages 495-508, XP000478665**
- **I.RIGOUTSOS ET AL: "Combinatorial Pattern Discovery in Biological Sequences: The TEIRESIAS Algorithm" BIOINFORMATICS, vol. 14, no. 1, 1998, pages 55-67, XP002094907**
- **COOK J E ET AL: "AUTOMATING PROCESS DISCOVERY THROUGH EVENT-DATA ANALYSIS" PROCEEDINGS OF THE 17TH. ANNUAL CONFERENCE ON SOFTWARE ENGINEERING, SEATTLE, APR. 23 - 30, 1995, no. CONF. 17, 23 April 1995, pages 73-82, XP000545655 ASSOCIATION FOR COMPUTING MACHINERY**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

EP 0 985 995 B1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 762

Description

Technical Field

[0001] This invention relates to intrusion detection, i. e. the detection of security problems in a computer network or on any computer within said network. It is particularly suited to detect outsiders trying to break into a computer system (e.g. via the net) and/or to detect insiders misusing the privileges they have received (e.g. someone internal reading confidential data that he/she is not entitled to). In brief, the invention uses a behavior-based approach for a pattern-oriented intrusion detection system.

Background of the Invention

[0002] Generally, an intrusion detection system dynamically monitors actions that are taken in a given environment and decides whether these actions are symptomatic of an attack or constitute a legitimate use of the environment.

[0003] Essentially, two main intrusion detection methods are known. The first method uses the knowledge accumulated about attacks and looks for evidence of their exploitation. This method is referred to as *knowledge-based*. The second method builds a reference model of the usual behavior of the system being monitored and looks for deviations from the observed usage. This method is referred to as *behavior-based*.

[0004] In the knowledge-based approach, the underlying assumption is that the system knows all possible attacks. There is some kind of a *signature* for each attack and the intrusion detection system searches for these signatures when monitoring the traffic. E.g., one may monitor the audit trails on a given machine, the packets going onto the net, etc. This first approach is addressed and described by Shieh et al. in US patent 5 278 901, which also gives a good overview over the technology. An advantage of this method is that no or only few false alarms are generated, i.e. the false alarm rate is low; the main disadvantage is that only those attacks can be located that are already known. Any newly developed intrusion attack would usually remain undetected since its signature is still unknown and thus the system does not search for it.

[0005] Unfortunately, there are nowadays so many attacks that the set of signatures is growing very fast. Also, some signatures are difficult to express and an algorithm to search for them can be rather time-consuming. Nevertheless, this approach has proven its usefulness and there are products using this approach available on the market: NetRanger by Cisco Systems, Inc., and RealSecure by Internet Security Systems, Inc., are two examples of such available products.

[0006] The second, the behavior-based, approach starts from the assumption that if an attack is carried out against a system, its "behavior" will change. Therefore, the approach is to define a kind of *normal profile* of a system and watch for any deviation from this defined normal profile. Different techniques can be applied (e. g. statistics, rule-based systems, neural networks) using different targets (e.g. the users of the system, the performances of the network, the CPU cycles, etc...). The main advantage of this method over the knowledge-based one is that the attacks do not need to be known in advance, i.e. that unknown attacks can be detected. Thus, the detection remains up-to-date without having to update some database of known signatures. But there are disadvantages: deviations can occur without any attack (e.g. changes in the activity of the user, new software installed, new machines, new users, etc.). Therefore, all known efforts in this direction have been facing a rather high rate of false alarms. There appears to be only one product on the market using this approach: CMDS by Science Applications International Corporation.

[0007] In "A Sense of Self for Unix Processes" by S. Forrest et al., Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 120-128, Oakland, California, May 1996, it is described how to model the behavior of the "*sendmail daemon*", i.e. a program running permanently in the background without user interaction, using the sequences of system calls that this program generates while running. The idea is to build a table of all the sequences of a given fixed length (here 5, 6, and 11) of consecutive system calls that could be found when watching such a *sendmail daemon* running. The claim was that if one tries to take advantage of a vulnerability in the *sendmail* code, then this would generate a sequence of systems calls not found in a "normal" table, i.e. a table generated from a sample with normal behavior. However, when experimenting with this approach, one discovers that the table necessary can become fairly large. It must be stressed that all the sequences of system calls in this table have the same length, i.e. lengths of 5, 6, and 11. It has been shown in "Fixed vs. Variable-Length Patterns for Detecting Suspicious Process Behavior" by H. Debar, M. Dacier, M. Nassehi, and A. Wespi, Proceedings of ESORICS 98, Louvain-la-Neuve, Belgium, September 1998, that when trying to find what the best length for the sequences is ("best" meaning producing the shortest table of patterns while covering all possible sequences) the result is that the "best" length is 1. This means that the system does not search for unseen sequences but for unseen system calls. The consequence is that if an attack does not use any unseen system call it will not be detected. This is generally unacceptable since it may be possible to run an attack without using a previously unseen system call.

[0008] There are two classes of information sources for intrusion detection systems as described in "Towards a Taxonomy of Intrusion Detection Systems" by H. Debar, M. Dacier, and A. Wespi, IBM Research Report 3030, June 1998. Based on the location from where the information can be retrieved, it is differentiated between

IPR2024-00027
CrowdStrike Exhibit 1002 Page 763

host-based and network-based intrusion detection systems. Examples of host-based information sources are the so-called *C2 audit trails,* the *syslog* files known in the UNIX operating system, or the *event logs* in Windows NT. Network-based information is mainly retrieved by analyzing the network packets.

[0009] As will be described in detail further below, the present invention relates to behavior-based intrusion detection using host-based information sources. For a given process, the intrusion detection system decides whether the process behavior can be judged as normal or abnormal. Abnormal behavior is an indication of an intrusion.

[0010] As mentioned above, Forrest et al. describe in "A Sense of Self for UNIX Processes", Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 120 - 128, Oakland, California, May 1996, a process model using a set of fixed-length patterns. These patterns correspond to all the possible patterns that can be found in the event sequences recorded during the training phase.

[0011] This poses a problem since a careful look at the sequences of audit events that can be generated by the so-called *ftp daemon* running under AIX shows that there are very long subsequences which repeat frequently. For example, many process instantiations start with an identical subsequence that has a length of 40 audit events. Thus, since the described fixed-length approach does not consider such a characteristic, any result of an intrusion detection method based on such a fixed-length approach is distorted and certain intrusions and/or misuses cannot be detected.

[0012] In "Intrusion Detection via System Call Traces" by A.P. Kosoresow and S.A. Hofmeyr, IEEE Software, pp. 35 - 42, Sep./Oct. 1997, it is shown that variable-length patterns can be used to model the normal behavior of a process. However, the patterns presented in this publication were constructed manually due to the lack of an automated method. It is obvious that such a manual selection or design of the patterns is inadequate for an automatic intrusion detection of the kind here approached.

[0013] A significantly different approach for a pattern-oriented intrusion detection system is disclosed in US Patent 5 278 901 to Shieh et al. It shows an intrusion detection system based on object privilege and information flow, i.e. does not use the deviation from a "typical" activity profile as described above. The approach by Shieh at al. is a knowledge-based intrusion detection system which is in contrast to the behavior-based approach of the present invention. Furthermore, the Shieh patent covers mainly the problem of detecting violations against previously defined access control policies while the present invention aims at detecting any type of attacks. The complexity of the solution chosen in the Shieh patent, however, makes this approach unsuitable to solve the problems which the present invention addresses.

[0014] To summarize, it is an object of the present invention to provide a simple and reliable method and apparatus for the detection of intrusions into a computer system, based on event patterns and particularly directed to detect deviations from a "normal" process behavior, and thus to detect attacks performed against said process. A more specific object is to generate, preferably automatically, so-to-speak "natural" patterns for the description of the process behavior and thus produce a very condensed resulting pattern table. Another specific object is to allow the use of highly efficient pattern matching algorithms, especially by producing a relatively small pattern matching table. A further specific object is to produce a pattern table with most representative patterns, independent of their length. A still further specific object is to produce a pattern table with less but longer entries than tables obtained with known approaches and thus improve the detection of attacks. A still further object is to define rules that specify when a deviation from the normal behavior is significant enough to raise an alarm.

Summary of the Invention

[0015] It appears that the idea of having sequences investigated is very important, but that building fixed length sequences is leading to unsatisfactory results, at least when these sequences do not exceed a certain minimal length. Therefore, the invention uses a new approach by focusing on a novel algorithm, the Teiresias algorithm, as described by I. Rigoutsos and A. Floratos in "Combinatorial Pattern Discovery in Biological Sequences - The TEIRESIAS Algorithm" in BioInformatics, pp. 55-67, Vol. 14, No. 1, 1998. This algorithm is also the subject of US patent 6 108 666 to Floratos et al.

[0016] The Teiresias algorithm, developed for a different purpose and never considered for intrusion detection, is used to search for *patterns*, i.e. all the subsequences that appear at least twice in a set of input sequences. Though there are other algorithms besides Teiresias that solve the problem of discovering all patterns, none of them is as efficient and fast as the Teiresias algorithm. Generally speaking, the results achieved with the Teiresias algorithm are far superior to anything produced by the prior art approaches.

[0017] A particular advantage is that, using the Teiresias algorithm, the longest patterns in a set of input sequences can be found. This is important since a table of long patterns appears to be more "representative" of a specific process than a table of short patterns. Since longer patterns usually contain more context information, it appears that they are more significant for a process than short patterns. On the other hand, short patterns are not necessarily unique for a specific process, but may appear in other processes. It is even possible that short patterns are part of an attack. The longer a pattern is, the lower is the probability that this pattern is part of other processes or even an attack. Consequent-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 764

ly, it was found that there are attacks that can be detected with the new technique according to the present invention, which attacks remained undetected with other techniques.

[0018] A further advantage is that, obviously, a small pattern table allows to implement efficient pattern matching algorithms and still works reasonably fast. Both advantages lead to an improved detection of attacks.

[0019] To summarize, the present invention provides a method and a system for reliably detecting intrusion patterns, thereby minimizing the probability of false alarms.

[0020] The method and apparatus for an intrusion detection system according to the invention, using the described variable-length approach when investigating event patterns, operates in two modes, a *training mode* and an *operation mode.*

[0021] In the training mode, generally speaking, the behavior of a process is defined based on the system events it generates. System events are either the system calls that are invoked by the process or the audit events generated on behalf of the process. The process model is a table of patterns, i.e. sequences or subsequences of events, which are representative of the process examined. To get a complete picture of the process, it is important that as many different event sequences as possible are generated and analyzed.

[0022] In this training mode, variable-length patterns are retrieved from the event sequences generated by or on behalf of the process. All events of a specific type generated from the invocation of the process until its end constitute an *event sequence.* Different process invocations may result in different event sequences. *Patterns* are subsequences of the event sequences; patterns that are characteristic for the process are stored in a *pattern table.* The pattern table represents the process model.

[0023] In the operation mode of the present invention, it is decided whether the event streams created on behalf of the process can be matched by the patterns in the pattern table, which corresponds to a normal process behavior, or whether there are subsequences of unmatched events. Unmatched events represent a deviation from the normal behavior and may thus indicate an intrusion or misuse, called an *attack.* Significant deviations result in raising an alarm.

[0024] As already mentioned, the present invention is advantageous because patterns are generated that are "natural" for the description of the process behavior. The use of variable-length patterns to build the set of representative patterns results in a pattern table with less, but longer entries than tables obtained with other approaches. As explained, longer patterns contain more context information and are therefore more representative for a particular process than short patterns. Furthermore, a small pattern table increases the speed of the detection process. It is obvious that, when looking for a pattern that matches part of a given sequence, searching in a small set of patterns is faster than in a large set. Therefore, small pattern tables allow to speed up the pattern matching process.

[0025] The invention is defined according to claims 1 (method) and 10 (apparatus).

Brief Description of the Drawings

[0026] The foregoing and other features and advantages of the invention will be apparent from the following more detailed description of a preferred embodiment of the invention, as illustrated in the accompanying drawings, in which:

- Figure 1 shows the components of an intrusion detection system according to the invention, based on the analysis of event patterns;
- Figure 2 is a sample output of the Event Recording component;
- Figure 3 is a sample output of the Process Filtering component;
- Figure 4 is a sample output of the Translation component;
- Figure 5 is a sample output of the Reduction and Aggregation component;
- Figure 6 shows the patterns as detected by the Teiresias algorithm for a set of sample input strings;
- Figure 7 is an illustration of the pattern reduction algorithm applied to the sample string set introduced in Figure 6;
- Figure 8 is an illustration of the pattern matching algorithm for a case where the input string can be covered; and
- Figure 9 is an illustration of the pattern matching algorithm for a case where the input string cannot be completely covered.

Detailed Description of the Invention

[0027] In the following first section, the components of the intrusion detection system are described. In a subsequent second section, the algorithms used in selected components are discussed.

1. The Intrusion Detection System and its Components

[0028] Figure 1 shows the components of the intrusion detection system. The system consists of two parts: an off-line part and an on-line part. The off-line part rep-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 765

resents the training phase or mode, and the on-line part the real operation or operation mode. In the training mode, a model of the normal behavior of the process examined is generated. In the operation mode, the instantiations of the process under the observation of the intrusion detection system are compared to the process model and, if a significant deviation is observed, an alarm 136 is raised.

[0029] A process execution 103 can trigger different types of events 104. Either one of the following two event sources can be used for the present invention:

- C2 audit events as they are recorded by the auditing system available on most UNIX and some other operating systems.
- System calls as they are recorded by programs coming with the operating system, e.g. strace, or by other system utilities.

[0030] It has to be noted that the two sources cannot be used interchangeably. Either audit events or system calls have to be considered. Any further use of the term *event* in this document may relate to audit events as well as system calls.

[0031] In the off-line part, it is possible to influence the process invocation in order to exercise as many different process execution paths as possible. For this purpose, we use the functional verification tests (FVT) as they are used by software developers to test all the different subcommands that can be executed by a process.

[0032] Other approaches would be to define manually a set of subcommands that are expected to cover all the process execution paths, or to just record the events of the process running in a real environment.

[0033] The events generated on behalf of a process are recorded by an event recording component 105. Event recording component 105 may not only record events by the process examined but also by other processes in the system. E.g. the auditing system does only allow to collect the audit events on a system level, i.e. either for all processes or for none. An event is described by several attributes, e.g. the process name, the event name, the process id, the parent process id, or the user id.

[0034] Event recording component 105 forwards the events to training system 102. Events are forwarded as triples comprising process name, event name, and process id, labeled 106. Figure 2 shows a sample of the events that are sent from event recording component 105 to a filtering component 107 in training system 102. Filtering component 107 first groups together the events belonging to the same process by keeping the chronological order of the events. All events belonging to the same process are called an *event sequence*. An event sequence consists of an unique *identifier* and a *list of events*. The events are given as tuples comprising the process and the event name. Not all event sequences are needed for the further processing. Only those sequences which are needed to analyze the behavior of the examined process are forwarded to a translation component 109.

[0035] Figure 3 shows the events of Figure 2 after applying the filtering component 107.

[0036] For the further internal processing, translation component 109 translates the event sequences into an internal data representation. Each event, i.e. the tuple consisting of the process and the event name, is translated into a single character of an alphabet Σ. The output of translation component 109 are strings of characters labeled 110.

[0037] The translation is bijective, i.e. identical events are translated into the same character, and a character is the translation of identical events only. The translation rules are generated on the fly and stored in a translation table 136. Figure 4 shows the result of the translation of the event sequences into strings.

[0038] The strings can be reduced further. A reduction and aggregation component 111 performs two tasks:

- Duplicate strings are removed.
- Consecutive occurrences of the same character are aggregated into a smaller number of the same character.

[0039] The first task results in a set of unique strings. Duplicate strings do not add any value to the further processing as will be seen later.

[0040] It can be observed that subsequences of ***N, N > >1,*** events are quite frequent, with ***N*** exhibiting small variations. An example is the *ftp login* session, where the *ftp daemon* closes several file handles inherited from the *inetd* process. Since the *inetd* process is not always in the same state, the number of its file handles may vary. As a consequence, the *ftp daemon* inherits not always the same number of file handles. Closing all the unneeded file handles results therefore in a varying number of file close operations.

[0041] There are two possible ways to aggregate characters:

- The identical consecutive characters are replaced with an extra, not yet used character.
- The ***N*** identical consecutive characters are comprised into ***M, 1 <= M <= N,*** characters.

[0042] The first approach increases the number of unique events and possibly also the number of patterns. Since the number of patterns should be kept small, the second approach with ***M = 1*** has been selected. The newly created strings have less semantics than the original ones, but no case is known where the character aggregation impacts the operation of the intrusion detection system.

[0043] The output of the reduction and aggregation component 111 are unique strings, labeled 112, where consecutive occurrences of the same character are re-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 766

moved. Figure 5 shows the strings of Figure 4 after being processed by reduction and aggregation component 111.

[0044] A pattern extraction component 113 determines the patterns which constitute the process model and stores them in a pattern table 135. The algorithms used to build pattern table 135 are explained in detail in subsequent sections.

[0045] Pattern table 135 is a key part of the intrusion detection system according to the invention. It links the off-line system 102 with the on-line system 122.

[0046] The on-line part has nearly the same components as the off-line part. However, a main difference is that the process to be examined is not under control of the intrusion detection system. In on-line system 122, the event recording component 125 and the process filtering component 127 are the same as in the off-line system. The translation component 129 is different with respect to the fact that audit events are translated based on the entries retrieved from a translation table 134. If there is an event for which no entry in translation table 129 exists, the event is translated to a dummy character. For each event sequence, it has to be decided whether there is a sign of an intrusion or not. Therefore, there is no reduction component (like component 111 in the off-line part) that removes duplicate sequences as in the training system. There is only an aggregation component 131.

[0047] The pattern matching component 133 receives its input strings 132 from the aggregation component 131. By applying the algorithm described in the subsequent section of this description, it is tried to match all the input strings with the patterns of pattern table 135. However, there may be strings that remain with uncovered characters. Depending on the number of consecutively uncovered characters, it is decided whether there is an indication of an intrusion, and whether an alarm 136 has to be issued.

[0048] For the *ftp daemon*, a threshold of 6 characters was selected, i.e. if there are 7 subsequent uncovered characters, an alarm is issued.

2. Algorithms

[0049] In this section, a sample algorithm to build the pattern table and a sample algorithm to cover the input stream are described. They represent the best algorithms we know so far. However, we can think of variations of these algorithms. For example, the algorithm to build the pattern table sorts the patterns based on the number of characters they can cover at the beginning and end of an input string. We can think of other sort criteria like the total number of characters covered by a pattern or the number of occurrences of a pattern.

2.1 Terminology and Notation

[0050] Consider a finite set of characters $\Sigma = \{c_1, c_2, ..., c_n\}$. The set Σ is called an alphabet. To denote a string of n, $n > 0$, identical consecutive characters $c \in \Sigma$, we write c^n. The term c^* denotes a string of identical consecutive characters of arbitrary length l, $l >= 0$. The term c^+ denotes a string of identical consecutive characters of arbitrary length m, $m > 0$. To denote an arbitrary string of length n, $n > 0$, we write $\{.\}^n$. $\{.\}^*$ denotes an arbitrary string of arbitrary length l, $l >= 0$, and $\{.\}^+$ denotes an arbitrary string of length m, $m > 0$.

[0051] The length of a string s is written as $|s|$. We write $c \in s$ if the character c is contained in the string s.

[0052] Given is a set of strings $S = \{s_1, s_2, ..., s_n\}$ over the alphabet Σ. A substring p that

- occurs at least twice in the set of strings S, and
- has a length $|p|$ of two or more characters

is called a pattern.

[0053] p^n denotes the pattern p repeated n, $n > 0$, times. p^* denotes the pattern p repeated l, $l >= 0$, times, and p^+ denotes the pattern p repeated m, $m > 0$, times.

[0054] A pattern p is maximal, if there is no pattern q for which holds:

- p is a substring of q with $|p| < |q|$, and
- the number of occurrences of the pattern $q \in S$ is equal or larger than the number of occurrences of the pattern $p \in S$.

[0055] A character $c \in s$ is said to be covered by the pattern p, if $c \in p$ and p is a substring of s.

[0056] A string s is said to be covered by a set of patterns P, if for each character c, $c \in s$, there is a pattern p, $p \in P$, so that c is covered by p.

[0057] A set of strings S is said to be covered by a set of patterns P, if each string s, $s \in S$, is covered by P. Additionally, P is said to cover S.

[0058] Given are a pattern p and a string s. Let us decompose the string s as follows:

$$s = p^l \{.\}^* p^r \; l, r >= 0$$

[0059] It is assumed that the decomposition is maximal, i.e. there are no l' and r' for which holds $l'+r' > l+r$.

[0060] The expression $(l + r) \cdot |p|$, i.e. the sum $l + r$ times the pattern length $|p|$, is called margin cover of the pattern p and the string s. It is written as $mCover(p,s)$. The margin cover of a pattern p and a string set $S = \{s_1, s_2, ..., s_n\}$, written as $mCover(p,S)$, is defined as

$$\sum_{i=1}^{n} mCover(p,s_i).$$

[0061] The total cover of a pattern p and a string set S, $tCover(p,S)$ is the total number of characters that can

IPR2024-00027
CrowdStrike Exhibit 1002 Page 767

be covered by the pattern p.

2.2 Determining the Set of Maximal Variable-Length Patterns

[0062] In a first step, all maximal patterns contained in the set S of input strings have to be determined. For this purpose, the Teiresias algorithm, as described by I. Rigoutsos and A. Floratos in "Combinatorial Pattern Discovery in Biological Sequences - The TEIRESIAS Algorithm" in BioInformatics, pp. 55-67, Vol. 14, No. 1, 1998, is used. A minimal pattern length m can be specified as argument for the Teiresias algorithm. The Teiresias algorithm will then only find the maximal patterns whose length is equal to or greater than this given minimal length.

[0063] Part a) of Figure 6 shows a sample input set of 3 strings and part b) shows the corresponding pattern set as discovered by the Teiresias algorithm (with m = 2). For each pattern, the total number of occurrences (first column) as well as the number of strings in which it occurs (second column) is given.

2.3 Reducing the Set of Patterns

[0064] Out of the set of patterns P consisting of all the maximal patterns found for the string set S, a subset of patterns R, $R \subset P$, is selected which covers S. As an example, the following algorithm can be used to build the reduced pattern set R.

1. Let m denote the minimal pattern length that was used to generate the set of maximal variable-length patterns. Add each s, $s \in S \wedge |s| < 2 \cdot m$, to P and remove it from S.
2. If $P = \emptyset$, then add all $s \in S$ to the reduced pattern set R and exit.
3. For each $p \in P$ calculate $mCover(p,S)$.
4. If there is a pattern p fulfilling $mCover(p,S) > 0$, then select a pattern r for which $mCover(r,S)$ is maximal, i.e. there is no pattern q for which holds:

$$mCover(q,S) > mCover(r,S) \text{ or}$$

$$mCover(q,S) = mCover(r,S) \wedge |q| > |r|.$$

5. Add r to the reduced pattern set R and remove it from P.
6. Remove first all the matching substrings adjacent to the beginning and end of a string, i.e. remove strings of the form $s = r^*$, and replace strings of the form $s = r^*s'$ or $s = r^*s''$ with s' or s'', respectively.
7. Remove then the matching substrings that are not adjacent to the beginning and end of a string, i. e. as long as there is a $s \in S$, $s = s'r^*s'' = \{.\}^v \{.\}^*r+\{.\}^*\{.\}^w$, m being the minimal pattern length and $v,w > 0$, replace s with the two strings s' and s''. v and w specify the minimal length of the resulting new strings s' and s'', respectively. Setting them equal to m enforces that all patterns have a minimal length m. However, different settings are possible.
8. If one of the strings s that have been newly added to S has a length $|s| < 2 \cdot m$, m being the minimal pattern length, remove s from the set of strings S and add it to the set of patterns P.
8. If $S \neq \emptyset$, go to step 2.

[0065] Figure 7 illustrates the pattern reduction algorithm applied to the sample string set introduced in Figure 6. For each reduction step, the string set, the pattern set, and the reduced pattern set are shown. For each pattern in the pattern set, its $mCover$ value is listed. The pattern with the highest $mCover$ value is moved to the reduced pattern set, and matching substrings are removed from the string set. In this example, not all patterns are needed to cover all the strings.

[0066] In the example of the *ftp daemon*, the Teiresias algorithm determines about 600 maximal variable-length patterns. After applying the reduction algorithm, about 50 patterns remain.

2.4 Pattern Matching

[0067] We describe a sample pattern matching algorithm. The algorithm tries to match the input stream by concatenating patterns, i.e. the patterns are placed one right after the other. A variation would be to allow overlapping patterns.

[0068] At certain points of the pattern matching process, there may be several patterns that match the input stream and it has to be decided which pattern to select. As an heuristic, a pattern is selected if a sequence of d, $d > 0$, patterns can be found that matches the input stream right after the pattern under consideration.

1. Set the counter of consecutively uncovered characters, u, to 0.
2. Wait until there are at least $k = d \cdot |p_{mean}|$ characters in the input stream where d is the parameter as explained in the introduction to this algorithm and $|p_{mean}|$ is the mean length of all patterns $p \in P$, or until the end of an input sequence has been reached.
3. Find a pattern $p \in P$ that covers the beginning of the input stream T. If no pattern can be found, go to step 6.
4. Find $d > 0$ patterns $q_1, q_2, ..., q_d$, so that the string $t = pq_1q_2...q_d$ covers the beginning of the stream. If there are e patterns $q_1, q_2, ..., q_e$, $0 < e < d$, that cover the whole input sequence, set $t = pq_1q_2...q_e$.

   (a) If t matches the whole input sequence, remove the input sequence and go to step 1.
   (b) If d patterns can be found covering the be-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 768

ginning of the input stream, remove the pattern ***p*** from the input stream and go to step 1.

5. Determine all pattern combinations that cover the beginning of the input stream. Select the pattern combination that covers the longest character sequence, remove it from the input stream, and go to step 1.
6. Skip one character and increase ***u*** by ***1.***
7. If ***u = n + 1, n*** being the threshold for the number of consecutively uncovered characters, raise an alarm.
8. Go to step 2.

[0069] Figures 8 and 9 are two illustrations of the pattern matching algorithm. One for the case of a fully-covered string, one for the case of a partially-covered string.

[0070] The pattern matching algorithm processes the pattern table found on top of Figures 8 and 9. In Figure 8, the matching algorithm first finds the pattern "ABC" matching the beginning of the sample string. Before accepting this pattern as a valid match, the pattern "ABC" must be validated be finding ***d=3*** patterns matching the remainder "ABCDXYZGHI" of the string. Since such three patterns can be found, namely "ABCD", XYZ", and "GHI", the pattern "ABC" is accepted and deleted from the sample string. Processing continues with the string "ABCDXYZGHI".

[0071] In Figure 9, we have again the same pattern table as in Figure 8. However, the sample string to be matched is somewhat different. Again, the pattern "ABC" is selected as candidate to match the beginning of the sample string. However, we cannot find ***d=3*** patterns that match the remainder "ABCDKMWHF" of the string. This implies that another pattern that matches the beginning of the sample string should be tried. Since there is no such other pattern, the pattern sequence has to be found that matches the longest portion of the sample string. Because the two pattern "ABC" and "ABCD" match the longest subsequence, they are selected. The characters "KMW" cannot be matched and are skipped. Processing continues with the string "HF".

[0072] A pattern-oriented, behavior-based, variable-length intrusion detection model was defined. The main advantage of this inventive model is that it generates a kind of "natural" patterns or signatures of the process to be monitored, which patterns very well represent the "normal" behavior. Thus, deviations - which indicate intrusion or misuse - can be easier detected than with previously known methods.

[0073] While the present invention has been particularly shown with reference to one specific embodiment, it is obvious to someone skilled in the art that it can be adapted to match the environment in which it is going to be used, whether for the detection of unauthorized transactions in the banking arena, of viruses in a computer network, of unauthorized entry to buildings with restricted access, or of unallowed data exchange between data bases, to name a few.

Claims

1. A method for detecting intrusion attempts in a computer or computer system, said method comprising in combination:

   - in a training mode, building a table of characteristic, process-constituting patterns defining normal behavior of a model process in said computer or computer system by performing the following steps:
     - building a first event sequence by filtering a first event stream generated by said model process,
     - by using the so-called Teiresias algorithm, extracting event sequence patterns from said first event sequence, said patterns constituting said model process,
     - storing said process-constituting patterns; and
   - in an operation mode, extracting characteristic patterns from an actual process by performing the following steps:
     - building a second event sequence by filtering an event stream generated by said actual process,
     - matching said second event sequence with said stored process-constituting patterns, and
     - indicating the result of said matching step.

2. The method for intrusion detection according to claim 1, wherein
   - in the training mode, the first event sequence is translated and the rules used for said translation are stored,
   - in the operation mode, the second event sequence is translated using said stored translation rules.

3. The method for intrusion detection according to claim 2, wherein the translation is a dynamic, on-the-fly translation.

4. The method for intrusion detection according to any

IPR2024-00027
CrowdStrike Exhibit 1002 Page 769

of the claims 1 to 3, wherein

- in the training mode, the first event sequence is compressed according to a given set of aggregation rules,
- in the operation mode, the second event sequence is compressed using said given set of aggregation rules.

5. The method for intrusion detection according to one or more of the preceding claims, wherein

- an event stream generated by either one or both of the processes is recorded and
- said recorded event stream filtered to build the first and/or second event sequence.

6. The method for intrusion detection according to one or more of the preceding claims, wherein the process-constituting patterns of the first event sequences contain patterns of varying lengths, in particular patterns of maximal lengths.

7. The method for intrusion detection according to one or more of the preceding claims, further including a reduction step in the training mode, whereby any duplications in the obtained event sequence are removed.

8. The method for intrusion detection according to one or more of the preceding claims, wherein the training mode is carried out under the control of the intrusion detection system.

9. The method for intrusion detection according to one or more of the claims 2 to 7, wherein all or part of the method steps are applied in the following sequences

- in the training mode:
  1. event recording,
  2. process filtering,
  3. translation and storage of translation rules,
  4. reduction and aggregation,
  5. pattern extraction and storage; and
- in the operation mode:
  1. event recording,
  2. process filtering,
  3. translation based on stored translation rules,
  4. aggregation,
  5. pattern matching with stored patterns.

10. An apparatus for detecting intrusion attempts in a computer or computer system, said apparatus comprising in combination:

- a first filtering component (107) for filtering, in a training mode branch, a first event stream generated by a model process (103) and building a first event sequence (108),
- a pattern extraction component (113) extracting event sequence patterns from said first event sequence by using the so-called Teiresias algorithm, said patterns constituting said model process,
- a pattern table component (135) storing said extracted, process-constituting patterns defining normal behavior of said model process,
- a second filtering component (127) for building, in an operation mode branch, a second event sequence by filtering an event stream generated by an actual process (123),
- a pattern matching component (133) for matching said second event sequence with said process-constituting patterns stored in said pattern table component (135), and
- an indicator component (136) for indicating the output of said matching component (133).

11. The apparatus for intrusion detection according to claim 10, further comprising

- a first translation component (109) which, in the training mode branch, translates the first event sequence,
- a translation table (134) for storing the translation rules used for said translation,
- a second translation component (129) which, in the operation mode branch, translates the second event sequence, using said translation rules stored in said translation table (134).

12. The apparatus for intrusion detection according to any of the claims 10 and 11, further comprising

IPR2024-00027
CrowdStrike Exhibit 1002 Page 770

- a first compression component (111) which, in the training mode branch, compresses the first event sequence according to a given set of aggregation rules, and
- a second compression component (131) which, in the operation mode branch, compresses the second event sequence using said given set of aggregation rules.

13. The apparatus for intrusion detection according to claim 12, further comprising a reduction component for removing duplicates in the event sequence obtained in the training mode branch, in particular a reduction component combined with the first compression component (111).

14. The apparatus for intrusion detection according to one or more of the claims 10 to 13, wherein all or part of the components are arranged in the following working sequences

- in the training mode branch:
  1. event recording component (105),
  2. process filtering component (107),
  3. translation component (109) and translation table (134),
  4. reduction and aggregation component (111),
  5. pattern extraction component (113) and pattern table (135); and
- in the operational mode branch:
  1. event recording component (125),
  2. process filtering component (127),
  3. translation component (129), connected to said translation table (134),
  4. aggregation component (131),
  5. pattern matching component (133), connected to said pattern table (135).

15. The apparatus for intrusion detection according to one or more of the preceding claims, wherein all or part of the components are arranged such that, to avoid duplication, they can be used alternatively either in the training mode branch or the operation mode branch.

Patentansprüche

1. Verfahren zur Detektion von Eindringversuchen in einem Rechner oder einem Rechnersystem, wobei das Verfahren eine Kombination der folgenden Modi und Schritte umfasst:
   - einen Trainingsmodus, in dem eine Tabelle charakteristischer, den Prozess darstellender Muster, die das normale Verhalten eines Modellprozesses in dem Rechner oder dem Rechnersystem definieren, durch Ausführen der folgenden Schritte erstellt wird:
     - Erstellen einer ersten Ereignissequenz durch Filtern eines ersten durch den Modellprozess generierten Ereignisdatenstroms,
     - Verwenden des so genannten Teiresias-Algorithmus zum Extrahieren von Ereignissequenzmustern aus der ersten Ereignissequenz, wobei die Muster den Modellprozess darstellen,
     - Speichern der den Prozess darstellenden Muster; und
   - einen Betriebsmodus, in dem aus einem realen Prozess charakteristische Muster durch Ausführen der folgenden Schritte extrahiert werden:
     - Erstellen einer zweiten Ereignissequenz durch Filtern eines durch den realen Prozess generierten Ereignisdatenstroms,
     - Vergleichen der ersten Ereignissequenz mit den gespeicherten, den Prozess darstellenden Mustern und
     - Anzeigen des Ergebnisses des Vergleichsschrittes.
2. Verfahren zur Eindringdetektion nach Anspruch 1, wobei
   - die erste Ereignissequenz im Trainingsmodus umgesetzt wird und die bei der Umsetzung angewendeten Regeln gespeichert werden,
   - die zweite Ereignissequenz im Betriebsmodus unter Verwendung der gespeicherten Umsetzungsregeln umgesetzt wird.
3. Verfahren zur Eindringdetektion nach Anspruch 2, wobei die Umsetzung eine dynamische Umsetzung ist und während der Verarbeitung stattfindet.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 771

4. Verfahren zur Eindringdetektion nach einem der Ansprüche 1 bis 3, wobei

   - die erste Ereignissequenz im Trainingsmodus nach einem vorgegebenen Satz von Komprimierungsregeln komprimiert wird,
   - die zweite Ereignissequenz im Betriebsmodus unter Verwendung des vorgegebenen Satzes von Komprimierungsregeln komprimiert wird.

5. Verfahren zur Eindringdetektion nach einem oder mehreren der vorangehenden Ansprüche, wobei

   - ein durch einen oder beide Prozesse generierter Ereignisdatenstrom aufgezeichnet und
   - der aufgezeichnete Ereignisdatenstrom gefiltert wird, um die erste und/oder zweite Ereignissequenz zu erstellen.

6. Verfahren zur Eindringdetektion nach einem oder mehreren der vorangehenden Ansprüche, wobei die den Prozess darstellenden Muster der ersten Ereignissequenzen Muster variabler Länge, insbesondere Muster maximaler Längen, enthalten.

7. Verfahren zur Eindringdetektion nach einem oder mehreren der vorangehenden Ansprüche, welches im Trainingsmodus ferner einen Reduzierungsschritt enthält, durch den alle doppelten Muster aus der erhaltenen Ereignissequenz entfernt werden.

8. Verfahren zur Eindringdetektion nach einem oder mehreren der vorangehenden Ansprüche, wobei der Trainingsmodus unter Kontrolle des Systems zur Eindringdetektion erfolgt.

9. Verfahren zur Eindringdetektion nach einem oder mehreren der Ansprüche 2 bis 7, wobei alle oder einige der Verfahrensschritte in den folgenden Sequenzen ausgeführt werden, und zwar

   - im Trainingsmodus:

     1. Aufzeichnen des Ereignisses,
     2. Filtern des Prozesses,
     3. Umsetzen und Speichern der Umsetzungsregeln,
     4. Reduzieren und Komprimieren,
     5. Extrahieren und Speichern der Muster; und

   - im Betriebsmodus:

     1. Aufzeichnen des Ereignisses,
     2. Filtern des Prozesses,
     3. Umsetzen auf Basis der gespeicherten Umsetzungsregeln,
     4. Komprimieren,
     5. Vergleichen der Muster mit den gespeicherten Mustern.

10. Vorrichtung zur Detektion von Eindringversuchen in einem Rechner oder einem Rechnersystem, wobei die Vorrichtung eine Kombination der folgenden Komponenten umfasst:

    - eine erste Filterkomponente (107) zum Filtern eines durch einen Modellprozess (103) generierten ersten Ereignisdatenstroms in einem Zweig des Trainingsmodus und zum Erstellen einer ersten Ereignissequenz (108),
    - eine Musterextrahierungs-Komponente (113) zum Extrahieren von Ereignissequenzmustern aus der ersten Ereignissequenz unter Verwendung des so genannten Teiresias-Algorithmus, wobei die Muster den Modellprozess darstellen,
    - eine Mustertabellen-Komponente (135) zum Speichern der den Prozess darstellenden extrahierten Muster, welche das normale Verhalten des Modellprozesses definieren,
    - eine zweite Filterkomponente (127) zum Erstellen einer zweiten Ereignissequenz in einem Zweig des Betriebsmodus durch Filtern eines durch einen realen Prozess (123) generierten Ereignisdatenstroms,
    - eine Mustervergleichs-Komponente (133) zum Vergleichen der zweiten Ereignissequenz mit den in der Mustertabellen-Komponente (135) gespeicherten, den Prozess darstellenden Muster, und
    - eine Anzeigekomponente (136) zum Anzeigen des Ausgabewertes der Vergleichskomponente (133).

11. Vorrichtung zur Eindringdetektion nach Anspruch 10, die ferner Folgendes umfasst:

    - eine erste Umsetzungskomponente (109), die im Zweig des Trainingsmodus die erste Ereignissequenz umsetzt,
    - eine Umsetzungstabelle (134) zum Speichern der beim Umsetzen angewendeten Umsetzungsregeln,
    - eine zweite Umsetzungskomponente (129), die im Zweig des Betriebsmodus die zweite Ereignisdatenfolge unter Anwendung der in der Umsetzungstabelle (134) gespeicherten Umset-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 772

zungsregeln umsetzt.

12. Vorrichtung zur Eindringdetektion nach einem der Ansprüche 10 und 11, wobei die Vorrichtung ferner Folgendes umfasst:

    - eine erste Komprimierungskomponente (111), die im Zweig des Trainingsmodus die erste Ereignissequenz gemäß einem vorgegebenen Satz von Komprimierungsregeln komprimiert, und
    - eine zweite Komprimierungskomponente (131), die im Zweig des Betriebsmodus die zweite Ereignissequenz unter Anwendung eines vorgegebenen Satzes von Komprimierungsregeln komprimiert.

13. Vorrichtung zur Eindringdetektion nach Anspruch 12, wobei die Vorrichtung ferner eine Reduzierungskomponente zum Entfernen doppelter Muster aus der im Zweig des Trainingsmodus erhaltenen Ereignissequenz und insbesondere eine Reduzierungskomponente in Kombination mit der ersten Komprimierungskomponente (111) umfasst.

14. Vorrichtung zur Eindringdetektion nach einem oder mehreren der Ansprüche 10 bis 13, wobei alle oder einige der Komponenten in den folgenden Arbeitssequenzen angeordnet sind:

    - im Zweig des Trainingsmodus:

        1. Ereignisaufzeichnungs-Komponente (105),
        2. Prozessfilter-Komponente (107),
        3. Umsetzungskomponente (109) und Umsetzungstabelle (134),
        4. Reduzierungs- und Komprimierungskomponente (111),
        5. Musterextrahierungs-Komponente (113) und Mustertabelle (135); und

    - im Zweig des Betriebsmodus:

        1. Ereignisaufzeichnungs-Komponente (125),
        2. Prozessfilter-Komponente (127),
        3. mit der Umsetzungstabelle (134) verbundene Umsetzungskomponente (129)
        4. Komprimierungskomponente (131),
        5. mit der Mustertabelle (135) verbundene Mustervergleichs-Komponente (133).

15. Vorrichtung zur Eindringdetektion nach einem oder mehreren der vorangehenden Ansprüche, wobei die Komponenten in der Vorrichtung so angeordnet sind, dass diese zur Vermeidung von Duplikaten alternativ entweder im Zweig des Trainingsmodus oder im Zweig des Betriebsmodus eingesetzt werden können.

Revendications

1. Un procédé de détection des tentatives d'intrusion dans un ordinateur ou un système d'ordinateur, ledit procédé comprenant, en combinaison :

    - 1en un mode d'apprentissage, la construction d'une table de caractéristiques, en constituant, par un processus, des motifs définissant le comportement normal d'un processus de modélisation dans l'ordinateur ou ledit système d'ordinateur, en exécutant les étapes suivantes :

        . construction d'une première séquence d'événements, par filtrage d'un premier flux d'événements généré par ledit processus de modélisation,

        . par utilisation ce que l'on appelle l'algorithme de Teiresias, extraction des motifs de séquence d'événements de ladite première séquence d'événements, lesdits motifs constituant ledit processus de modélisation,

        . stockage desdits motifs constituant le processus ; et

    - 2en un mode opérationnel, extraire des motifs de caractéristique d'un processus réel, en exécutant les étapes suivantes :

        . construction d'une deuxième séquence d'événements par filtrage d'un flux d'événements généré par ledit processus réel,

        . mise en coïncidence de ladite deuxième séquence d'événements avec lesdits motifs constituant le processus, ayant été stockés, et

        . indiquer le résultat de ladite étape de mise en coïncidence.

2. Le procédé de détection d'intrusion selon la revendication 1, dans lequel :

    . en mode d'apprentissage, la première séquence d'événements est traduite et les règles utilisées pour ladite traduction sont stockées,

    . en mode opérationnel, la deuxième séquence

IPR2024-00027
CrowdStrike Exhibit 1002 Page 773

d'événements est traduite en utilisant lesdites règles de traduction stockées.

3. Le procédé de détection d'intrusion selon la revendication 2, dans lequel la traduction est une traduction dynamique, au vol.

4. Le procédé de détection d'intrusion selon l'une quelconque des revendications 1 à 3, dans lequel :

   . en mode d'apprentissage, la première séquence d'événements est comprimée selon un jeu donné de règles d'agrégation,

   . en mode fonctionnel, la deuxième séquence d'événements est comprimée en utilisant ledit jeu donné de règles d'agrégation.

5. Le procédé de détection d'intrusion selon l'une ou plusieurs des revendications précédentes, dans lequel :

   . un flux d'événements, généré par l'un ou les deux processus, est enregistré et

   . ledit flux d'événements enregistré est filtré pour construire la première et/ou la deuxième séquence d'événements.

6. Le procédé de détection d'intrusion selon l'une ou plusieurs des revendications précédentes, dans lequel les motifs constituant le processus des premières séquences d'événements contiennent des motifs de différentes longueurs, en particulier des motifs de longueurs maximales.

7. Le procédé de détection d'intrusion selon l'une ou plusieurs des revendications précédentes, comprenant en outre une étape de réduction en mode d'apprentissage, de manière à supprimer d'éventuelles duplications dans la séquence d'événements obtenue.

8. Le procédé de détection d'intrusion selon l'une ou plusieurs des revendications précédentes, dans lequel le mode d'apprentissage est effectué sous la commande du système de détection d'intrusion.

9. Le procédé de détection d'intrusion selon l'une ou plusieurs des revendications 2 à 7, dans lequel la totalité ou une partie des étapes du procédé sont appliquées dans les ordres de succession suivants :

   - 3en mode d'apprentissage :

     - 4 enregistrement d'événement,

     - 5 filtrage de processus

     - 6 traduction et stockage des règles de traduction,

     - 7 réduction et agrégation,

     - 8 extraction et stockage de motif ; et

   - 9en mode opérationnel :

     o1 enregistrement d'événement,

     o2 filtrage de processus,

     o3 traduction basée sur des règles de transaction stockées,

     o4 agrégation

     o5 mise en coïncidence des motifs avec des motifs stockés.

10. Le procédé de détection des tentatives d'intrusion dans un ordinateur ou un système d'ordinateur, ledit dispositif comprenant, en combinaison :

    . un premier composant de filtrage (107) pour filtrer, en une branche de mode d'apprentissage, un flux d'événements généré par un processus de modélisation (103), et construction d'une première séquence d'événements (108),

    . un composant d'extraction de motif (113), extrayant des motifs de séquence d'événements de ladite première séquence d'événements, par utilisation de ce que l'on appelle l'algorithme de Teiresias, lesdits motifs constituant ledit processus de modélisation,

    . un composant de table de motif (135), stockant lesdits motifs extraits, constituant les processus, définissant le comportement normal dudit processus de modélisation,

    . un deuxième composant de filtrage (127), pour construire, dans une branche de mode opérationnel, une deuxième séquence d'événements par filtrage d'un flux d'événements généré par un processus réel (123),

    . un composant de mise en coïncidence de motifs (133), pour mettre en coïncidence, ladite deuxième séquence d'événements avec lesdits motifs de constitution de processus, stockés dans ledit composant de table de motifs (135), et

IPR2024-00027
CrowdStrike Exhibit 1002 Page 774

. un composant indicateur (136), pour indiquer le résultat dudit composant de mise en coïncidence (133).

11. Le dispositif de détection des tentatives d'intrusion selon la revendication 10, comprenant en outre :

. un premier composant de traduction (109) qui, dans la branche de mode d'apprentissage, traduit la première séquence d'événements,

. une table de traduction (134), pour stocker les règles de transaction utilisées pour ladite traduction,

. un deuxième composant de traduction (129), qui, dans la branche de mode opérationnel, traduit la deuxième séquence d'événements en utilisant lesdites règles de transaction stockées dans ladite table de transaction (134).

12. Le dispositif de détection des tentatives d'intrusion selon l'une quelconque des revendications 10 et 11, comprenant en outre :

. un premier composant de compression (111) qui, dans la branche de mode d'apprentissage, comprime la première séquence d'événements selon un jeu donné de règles d'agrégation, et

. un deuxième composant de compression (131) qui, à la branche de mode opérationnel, comprime la deuxième séquence d'événements, par utilisation dudit jeu donné de règles d'agrégation.

13. Le dispositif de détection d'intrusion selon la revendication 12, comprenant en outre une composante de réduction, pour supprimer les doubles dans la séquence d'événements obtenue dans la branche de mode d'apprentissage, en particulier un composant de réduction combiné au premier composant de compression (111).

14. Le dispositif de détection d'intrusion selon l'une ou plusieurs des revendications 10 à 13, dans lequel la totalité ou une partie des composants sont agencés dans les séquences de travail suivantes :

- 10en branche de mode d'apprentissage :
  - 11composant d'enregistrement d'événement (105),
  - 12composant de filtrage de processus (107),
  - 13composant de traduction (109) et table de traduction (134),
  - 14composant de réduction et d'agrégation (111),
  - 15composant d'extraction de motif (113) et table de motif (135) ; et
- 16dans la branche de mode opérationnel :
  - 1. composant d'enregistrement d'événement (125),
  - 17composant de filtrage de processus (127),
  - 18composant de traduction (129), relié à ladite table de traduction (134),
  - 19composant d'agrégation (131),
  - 20composant de coïncidence de motifs (133) relié à ladite table de motif (135).

15. Le dispositif de détection d'intrusion selon l'une ou plusieurs des revendications précédentes, dans lequel la totalité ou une partie des composants sont agencés de manière que, pour éviter toute duplication, ils puissent être utilisés de façon alternée, soit dans la branche de mode d'apprentissage, soit dans la branche de mode opérationnel.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 775

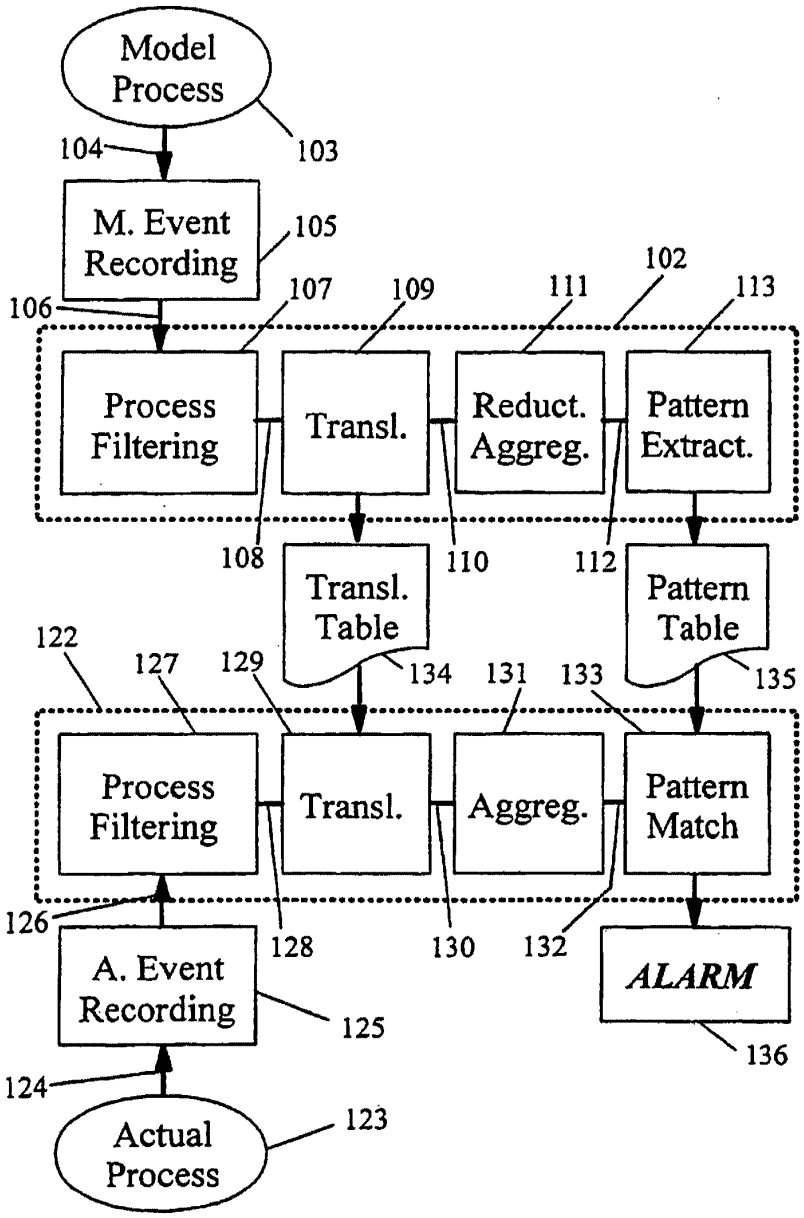


Fig.1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 776

Process	*Event*	*Process Id*
...		
ftpd	FILE_Close	16415
ls	PROC_Execute	16415
ls	FILE_Close	16415
fingerd	PROC_Execute	18210
ls	PROC_Delete	16415
fingerd	PROC_SetSignal	18210
ftpd	PROC_Create	18303
ftpd	FILE_Close	18303
ftpd	FILE_Close	18303
ftpd	FILE_Close	18303
fingerd	FILE_Read	18210
fingerd	PROC_Create	18210
ftpd	FILE_Close	18303
ftpd	PROC_SetSignal	18303
ftpd	FILE_Read	18303
ftpd	FILE_Read	18303
ftpd	FILE_Write	18303
ftpd	PROC_Delete	18303
fingerd	PROC_Execute	19415
fingerd	PROC_SetSignal	19415
fingerd	FILE_Read	19415
fingerd	PROC_Create	19415

Fig. 2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 777

0: (ftpd, FILE_Close), (ls, PROC_Execute),
(ls, FILE_Close), (ls, PROC_Delete)

1: (fingerd, PROC_Execute),
(fingerd, PROC_SetSignal), (fingerd, FILE_Read),
(fingerd, PROC_Create)

2: (ftpd, PROC_Create), (ftpd, FILE_Close),
(ftpd, FILE_Close), (ftpd, FILE_Close),
(ftpd, FILE_Close), (ftpd, PROC_SetSignal),
(ftpd, FILE_Read), (ftpd, FILE_Read),
(ftpd, FILE_Write), (ftpd, PROC_Delete)

3: (fingerd, PROC_Execute),
(fingerd, PROC_SetSignal),
(fingerd, FILE_Read), (fingerd, PROC_Create)

Fig. 3

0: A B C D
1: E F G H
2: I A A A A J K K L M
3: E F G H

Fig. 4

IPR2024-00027
CrowdStrike Exhibit 1002 Page 778

0: A B C D
1: E F G H
2: I A J K L M

Fig. 5

a)
0: A B C D E A
1: B C F D E A B C D
2: B C E A D E F D E

b)
4 3 B C
4 3 D E
3 3 E A
2 2 A B C D
2 2 D E A
2 2 F D E

Fig. 6

IPR2024-00027
CrowdStrike Exhibit 1002 Page 779

	Strings	*Pattern set*		*Reduced pattern set*
1)	ABCDEA	BC	4	---
	BCFDEABCD	DE	2	
	BCEADEFDE	EA	2	
		ABCD	8	
		DEA	3	
		FDE	3	
2)	EA	BC	4	ABCD
	BCFDE	DE	4	
	BCEADEFDE	EA	2	
		DEA	0	
		FDE	6	
3)	EA	BC	4	ABCD
	BC	DE	2	FDE
	BCEADE	EA	2	
		DEA	0	
4)	EA	DE	2	ABCD
	EADE	EA	4	FDE
		DEA	0	BC
5)	DE	DE	2	ABCD
		DEA	0	FDE
				BD
				EA
6)	---	DEA		ABCD
				FDE
				BD
				EA
				DE

Fig. 7

IPR2024-00027
CrowdStrike Exhibit 1002 Page 780

Table: ABCD XYZD ABC GHI XYZ HF
String: ABCABCDXYZGHI *ABC Selectable*
~~ABCD~~ ~~XYZD~~ ABC
Remainder of string: ABCDXYZGHI
ABCD
Remainder of string: XYZGHI
~~ABCD~~ ~~XYZD~~ ~~ABC~~ ~~GHI~~ XYZ
Remainder of string: GHI
~~ABCD~~ ~~XYZD~~ ~~ABC~~ GHI *ABC Validated*

Fig. 8

Table: ABCD XYZD ABC GHI XYZ HF
String: ABCABCDKMWHF *ABC Selectable*
~~ABCD~~ ~~XYZD~~ ABC
Remainder of string: ABCDKMWHF
Possible combinat.: ABC/ABC --> Length 6
ABC/ABCD --> Length 7 --> *Selected*

K	--->	Shifted	u = 1
M	--->	Shifted	u = 2
W	--->	Shifted	u = 3
HF			u = 0

~~ABCD~~ ~~XYZD~~ ~~ABC~~ ~~GHI~~ ~~XYZ~~ HF

Fig. 9

IPR2024-00027
CrowdStrike Exhibit 1002 Page 781

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 1/00 — A1

(11) International Publication Number: **WO 00/02115**

(43) International Publication Date: 13 January 2000 (13.01.00)

(21) International Application Number: PCT/US99/12181

(22) International Filing Date: 2 June 1999 (02.06.99)

(30) Priority Data:
09/109,866 6 July 1998 (06.07.98) US

(71) Applicant *(for all designated States except US)*: PRC INC. [US/US]; 1500 PRC Drive, McLean, VA 22102-5050 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: WALKER, Jeffrey, H. [US/US]; 1008 Bryn Mawr Drive, Papillon, NE 68128 (US).

(74) Agent: BERNER, Kenneth, M.; Lowe Hauptman Gopstein Gilman & Berner, LLP, Suite 310, 1700 Diagonal Road, Alexandria, VA 22314 (US).

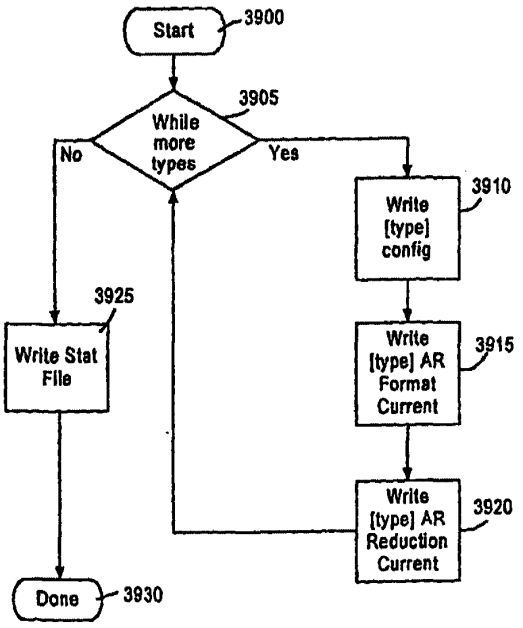
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD AND SYSTEM FOR REDUCING THE VOLUME OF AUDIT DATA AND NORMALIZING THE AUDIT DATA RECEIVED FROM HETEROGENEOUS SOURCES

(57) Abstract

A method of reducing the volume of native audit data from further analysis by a misuse and intrusion detection engine is disclosed. Typically, more than ninety percent of the volume of audit information received from heterogeneous operating systems does not need to be analyzed by a misuse and intrusion detection engine because this audit information can be filtered out as not posing a security threat. Advantageously, by reducing (eliminating) the volume of audit information, a misuse and intrusion engine can more quickly determine whether a security threat exists because the volume of data that the engine must consider is drastically reduced. Also, advantageously, the audit information that is forwarded to the engine is normalized to a standard format, thereby reducing the computational requirements of the engine. The method of reducing the volume of native audit data includes comparing each of the native audits against at least one template and against at least one native audit. By matching the native audits against templates of native audits that do not pose security threats, the native audits that do not pose security threats can be reduced out from further consideration. The native audits that are determined to pose potential security threats are transformed into a standardized format for further analysis by a misuse and intrusion detection engine.



IPR2024-00027
CrowdStrike Exhibit 1002 Page 782

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND SYSTEM FOR REDUCING THE VOLUME OF AUDIT DATA AND NORMALIZING THE AUDIT DATA RECEIVED FROM HETEROGENEOUS SOURCES

Field of the Invention

The present invention relates generally to intrusion detection systems for computer systems, and more particularly, to a method and apparatus for analyzing audit data received from heterogeneous sources, reducing the volume of the analyzed audit data from further consideration, normalizing audit trail records received from the heterogeneous sources to a standardized format, and performing an initial review of the normalized audit trail records. Based upon the initial review, intrusions can be detected and alerts provided.

Background of the Invention

The development of the computer and its astonishingly rapid improvement have ushered in the Information Age that affects almost all aspects of commerce and society. Just like the physical infrastructures that support the American economy, there is a highly developed computer infrastructure that supports the American and worldwide economy.

Besides traditional physical threats to United States security, the security of the United States is also dependent on protecting the computer infrastructure that supports American government and industry. The computer infrastructure is open to attack by hackers and others, who could potentially wreak havoc.

The President of the United States has recognized the existence of these infrastructures and has created the President's Commission on Critical Infrastructure Protection. This Commission was constituted to determine which

IPR2024-00027
CrowdStrike Exhibit 1002 Page 784

industries are critical and whether these industries were vulnerable to cyber attack. The Commission issued a report and deemed transportation, oil and gas production and storage, water supply, emergency services, government services, banking and finance, electrical power and telecommunications to be critical infrastructures which rely on the computer infrastructure.

A personal computer and a modem with access to the Internet are all the tools that a computer hacker needs to conduct a cyber attack on a computer system. The rapid growth of a computer-literate population ensures that millions of people possess the skills necessary to consider a cyber attack. The computer literate population includes recreational hackers who attempt to gain unauthorized electronic access to information and communication systems. These computer hackers are often motivated only by personal fascination with hacking as an interesting game. Criminals, and perhaps organized crime, might also attempt personal financial gain through manipulation of financial or credit accounts or stealing services. Industrial espionage can also be the reason for a cyber attack on a competitor's computer system. Terrorists may attempt to use the computer infrastructure. Other countries may use the computer infrastructure for national intelligence purpose. Finally, there is the prospect of information warfare, which is a broad, orchestrated attempt to disrupt a United States military operation or significant economic activity.

A typical secure computer network has an interface for receiving and transmitting data between the secure network and computers outside the secure network. A plurality of network devices are typically behind the firewall. The interface may be a modem or an Internet Protocol (IP) router. Data received by the modem is sent to a firewall which is a network security device that only allows data packets from a trusted computer to be routed to specific addresses within the secure computer network. Although the typical firewall is adequate to prevent outsiders from accessing a secure network, hackers and others can often breach a firewall. This can occur by cyber attack where the firewall becomes overwhelmed with requests and errors are made permitting access to an

IPR2024-00027
CrowdStrike Exhibit 1002 Page 785

unauthorized user. As can be appreciated, new ways of overcoming the security devices are developed everyday. An entry by an unauthorized computer into the secured network, past the firewall, from outside the secure network is called an intrusion. This is one type of unauthorized operation on the secure computer network.

Another type of unauthorized operation is called a misuse. A misuse is an unauthorized access by a computer within the secure network. In a misuse situation, there is no breach of the firewall. Instead, a misuse occurs from inside the secure computer network. A misuse can be detected when an authorized user performs an unauthorized, or perhaps, infrequent operation which may raise the suspicion that the authorized user's computer is being misused. For example, an unauthorized user could obtain the password of an authorized user and logon to the secured network from the authorized computer user's computer and perform operations not typically performed by the authorized user. Another example might be where a terrorist puts a gun to the head of an authorized user and directs the authorized user to perform unauthorized or unusual operations.

There are systems available for determining that a breach of computer security has occurred. These systems can broadly be termed intrusion detection systems. Existing intrusion detection systems can detect intrusions and misuses. The existing security systems determine when computer misuse or intrusion occurs. Computer misuse detection is the process of detecting and reporting uses of processing systems and networks that would be deemed inappropriate or unauthorized if known to responsible parties. An intrusion is an entry to a processing system or network by an unauthorized outsider.

Processing system misuse detection and reporting research has been funded by U.S. government agencies that have concerns for the confidentiality of their computer systems. Researchers have generally been associated with large research organizations or national laboratories. These institutions have required detailed knowledge of technical computer security, known threats and vulnerabilities, protection mechanisms, standard operational procedures,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 786

communications protocols, details of various systems' audit trails, and legal investigation of computer crimes. This misuse detection and reporting research has followed two basic approaches: anomaly detection systems and expert systems.

Anomaly detection systems look for statistically anomalous behavior. These systems assume that intrusions and other security problems are rare and that they appear unusual when compared to other user behavior. D. Denning, "An Intrusion Detection Model," Proc 1986 IEEE Symp. Security & Privacy (April 1986) provides an anomaly detection model (hereinafter the "Denning Model") for detecting intrusions into computer systems. The Denning Model uses statistical scenarios for user, dataset, and program usage to detect "exceptional" use of the system.

There are variations of the Denning Model and different applications of these various models. Anomaly detection techniques such as those based on the Denning Model, however, have generally proven to be ineffective and inefficient. Anomaly detection techniques, for instance, do not detect most actual misuses. The assumption that computer misuses would appear statistically anomalous has been proven false. When scripts of known attacks and misuses are replayed on computers with statistical anomaly detection systems, few if any of the scripts are identified as anomalous. This occurs because the small number of commands in these scripts are insufficient to violate profiling models.

In general, anomaly detection techniques cannot detect particular instances of misuses unless the specific behaviors associated with those misuses also satisfy statistical tests without security relevance. Anomaly detection techniques also produce false alarms. Most of the reported anomalies are purely statistical and do not reflect security problems. These false alarms often cause system managers to resist using anomaly detection method because they increase the processing system workload without substantial benefits.

Another limitation with anomaly detection approaches is that users activities are often too varied for a single scenario and can result in many false

IPR2024-00027
CrowdStrike Exhibit 1002 Page 787

alarms. Statistical measures also are not sensitive to the order in which events occur, and this may prevent detection of serious security violations that exist when events occur in a particular order. Scenarios that anomaly detection techniques use also may be vulnerable to conscious manipulation by users. Consequently a knowledgeable perpetrator may train the thresholds of detection system adaptive scenarios to accept aberrant behaviors as normal. Furthermore, statistical techniques that anomaly detection systems use require complicated mathematical calculations and, therefore, are usually computationally expensive.

Expert systems (also known as rule-based systems or production system) have had some use in misuse detection, generally as a layer on top of anomaly detection systems for interpreting reports of anomalous behavior. Since the underlying model was anomaly detection, they have the same drawbacks of anomaly detection techniques.

Expert system approaches, in addition, are themselves inherently inefficient. S. Snapp, et al., "DIDS (Distributed Intrusion Detection System)" Proc. 14th Nat'l Computer Security Conf., Washington, D.C. (October 1991) describes one example of an expert system signature analysis model that detects misuse by looking for one specific event within a specific system context. In one study, this detection system was found to be two and four orders of magnitude slower than "hard-wired" techniques and much too slow for real-time operation. This also makes it impractical to use these systems to detect and report misuses of multiple associated processing systems through operation of a single misuse detection and reporting system.

Expert systems approaches are also not deterministic. Consequently, these rules are expressed in a declarative, non-procedural fashion. When rule changes occur, it is generally extremely difficult to predict how the new system will behave. This makes development and testing more complex and expensive. Moreover, expert system approaches are limited to the knowledge of the expert who programmed the rules into the system. However, an expert is only capable of programming the rules of behavior that the expert knows. Since there are often

IPR2024-00027
CrowdStrike Exhibit 1002 Page 788

many different paths to a particular misuse, the expert will be unable to create rules that represent all of these paths.

More recent attempts at detecting misuse have relied on a signature mechanism with a signature being the set of events and transitions functions that define the sequence of actions that form a misuse. A misuse engine that uses this signature mechanism is described in detail in U.S. patent no. 5,557,742. The signature mechanism uses audit trail records typically generated by computer operating systems. The user selects a plurality of misuses that together form the signature mechanism. Although the signature mechanism goes a step beyond expert systems, it is similar to an expert system because it relies upon signatures or rules.

A need exists for an intrusion detection system which can provide early warning of potential misuses and intrusions without relying on particular rules or signatures which can be easily thwarted. Early warning can be provided by eliminating most of the audit trail records before a misuse and intrusion detection engine further analyzes the audit trail records.

Summary of the Invention

It is, therefore, an object of the present invention to provide a method and apparatus for eliminating audit trail records from further consideration by an intrusion and misuse detection engine.

It is a further object of the present invention to normalize a portion of the audit trail records received from heterogeneous sources for further analysis by the misuse and intrusion detection engine.

A further object is to provide an early warning of a potential misuse or intrusion.

Another object is to use a weighting system using barriers and boundaries so that a potential misuse or intrusion is identified before it occurs.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 789

An additional object is to provide an intrusion detection system in which particular misuses do not have to be known in advance and which does not require rule sets or signatures of particular misuses.

These and other objects of the present invention are achieved by a method of reducing the volume of native audit data from further analysis by a misuse and intrusion detection engine. Typically, more than ninety percent of the volume of audit information received from heterogeneous operating systems does not need to be analyzed by a misuse and intrusion detection engine because this audit information can be filtered out as not posing a security threat. Advantageously, by reducing (eliminating) the volume of audit information, a misuse and intrusion engine can more quickly determine whether a security threat exists because the volume of data that the engine must consider is drastically reduced. Also, advantageously, the audit information that is forwarded to the engine is normalized to a standard format, thereby reducing the computational requirements of the engine. The method of reducing the volume of native audit data includes comparing each of the native audits against at least one template and against at least one native audit. By matching the native audits against templates of native audits that do not pose security threats, the native audits that do not pose security threats can be reduced out from further consideration. The native audits that are determined to pose potential security threats are transformed into a standardized format for further analysis by a misuse and intrusion detection engine.

The foregoing objects are also achieved by a method of reducing the volume of native audits received from at least one operating system with each of the native audits being in a particular format. The particular format is identified for each of the received native audits. Each of the received identified native audits are compared against at least one template and it is determined if each of the native audits matches at least one template. Each of the matched native audits is reduced.

The foregoing objects are also achieved by an article including at least one sequence of machine executable instruction on a medium bearing the executable

IPR2024-00027
CrowdStrike Exhibit 1002 Page 790

instructions in machine readable form. Execution of the instructions by one or more processors causes the one or more processors to identify the particular format for each of the received native audits. The one or more processors compares each of the received identified native audits against at least one template and determines if each of the native audits matches at least one template. The one or more processors reduces each of the matched native audits.

The foregoing objects are also achieved by a computer architecture for reducing the volume of native audits received from at least one operating system with each of the native audits being of particular format. The computer architecture includes identifying means for identifying a particular format for each of the received native audits. Comparing means are provided for comparing each of the received identified native audits against at least one template and determining if each of the native audits matches at least one template. Reducing means are provided for reducing each of the native audits.

The foregoing objects are also achieved by a computer system including a processor and a memory coupled to the processor, the memory having stored therein sequences of instructions, which, when executed by the processor, causes the processor to perform the steps of identifying a particular format for each of the received identified native audits. The processor performs the step of comparing each of the received identified native audits against at least one template. The processor determines if each of the native audits matches at least one template and reduces each of the matched audits.

Still other objects and advantage of the present invention will become readily apparent to those skilled in the art from following detailed description, wherein the preferred embodiments of the invention are shown and described, simply by way of illustration of the best mode contemplated of carrying out the invention. As will be realized, the invention is capable of other and different embodiments, and it several details are capable of modifications in various obvious respects, all without departing from the invention. Accordingly, the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 791

drawings and description thereof are to be regarded as illustrative in nature, and not as restrictive.

Brief Description of the Drawings

The present invention is illustrated by way of example, and not by limitation, in the figures of the accompanying drawings, wherein elements having the same reference numeral designations represent like elements throughout and wherein:

Figure 1 is a high-level block diagram of an exemplary secured computer network on which the present invention can be used;

Figure 2 is a high-level block diagram of an exemplary computer system with which the present invention can be used;

Figure 3 is a high-level block diagram of the hardware used for an audit reduction agent;

Figure 4 is an illustration of a logical architecture used for the audit reduction agent;

Figure 5A is an illustration of a native audit;

Figure 5B is an illustration of audits is a diagram of an embodiment of the invention as used in several fleets of trucks in a wireless network;

Figure 6 is an illustration of a matrix of templates;

Figures 7A-7C are illustrations of simple, simple-complex and complex-complex comparisons and interpretations, respectively;

Figure 7D is a flow diagram of the reduction process of the present invention; and

Figures 8-39 are flow diagrams of the process used for analyzing and reducing audit trail records from heterogeneous sources according to the present invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 792

Best Mode for Carrying Out the Invention

A method and apparatus for normalizing audit trail records received from heterogeneous sources to a standardized format, performing an initial review of the normalized audit trail records and reducing the number of audit trail records to be stored for later review are described. Based upon the initial review, intrusions can be detected and alerts provided. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention. The present invention is usable in conjunction with a patent application entitled "Dynamic System Defense for Information Warfare" assigned to the instant assignee and filed on even date herewith and incorporated by reference into this specification in its entirety.

To a great extent, monitoring of user and process behavior (and detection of anomalies in that behavior) can be based on security-relevant audit information. Audited systems produce a huge amount of audit data which can be analyzed and interpreted to determine whether security breaches are occurring or have occurred. Analysis allows an Information Security Officer (ISO) to recognize suspicious behavior and to respond to the behavior quickly enough to effectively counter a possible security threat. The move towards distributed data handling systems requires security monitoring to ascertain a system wide security posture as opposed to determining that there is a potential security threat for a single computer. This means that a hacker may attack two distributed computers simultaneously and the ISO needs to be aware that a security threat exists. The ISO can then identify when people inadvertently or purposely exploit the systems security weaknesses. The present invention addresses the following specific needs:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 793

1) reduces the volume of security relevant audit data,

2) performs timely audit analysis,

3) automates detection of suspicious behavior,

4) works with minimal user involvement,

5) helps define a model for security monitoring, and

6) allows an organization to respond to an evolving variety of threats.

Figure 1 is a block diagram illustrating an exemplary computer network 100 including a plurality of network devices on which an embodiment of the invention may be implemented. According to the present invention, an audit agent can reside on one or all of the nodes depicted on the computer network 100. As explained below, the audit agent resides on audit server 120 and receives audit trail information from each of the nodes on the computer network 120. The network devices include devices such as hosts, servers and personal computers. The present invention is usable on such networks as ARCnet, Ethernets and Token-Ring networks, wireless networks, among other networks. The network 100, in this example, has a central network cable 102, also known as media, which may be of any known physical configuration including unshielded twisted pair (UTP) wire, coaxial cable, shielded twisted pair wire, fiber optic cable, and the like. Alternatively, the network devices could communicate across wireless links.

The network 100 includes a network server 104 coupled to the network cable 102 and another server 106 coupled to the network cable 102. A host computer 108 is coupled to the network cable 102. A terminal 110 is coupled to the network cable 102. A personal computer 112 is coupled to the network cable 102. Each network device 104, 106, 108, 110, 112 can also be considered a node because each device has an addressable interface on the network. As can be appreciated, many other devices can be coupled to the network including

IPR2024-00027
CrowdStrike Exhibit 1002 Page 794

additional personal computers, mini-mainframes, mainframes and other devices not illustrated or described which are well known in the art.

A security server 114 is coupled to the network cable 102. A firewall 116 connects the secure network 100 to an interface 118. The firewall 116 is a combination hardware and software buffer that is between the internal network 100 and external devices outside the internal computer network 100. The network devices within the internal network 100 appear within the dashed lines in Figure 1, and the external devices outside the internal network appear outside the dashed lines in Figure 1. The firewall 116 allows only specific kinds of messages from external devices to flow in and out of the internal network 100. As is known, firewalls are used to protect the internal network 100 from intruders or hackers who might try to break into the internal network 100. The firewall 116 is coupled to an interface 118. The interface 118 is external to the network 100 and can be a modem or an Internet Protocol (IP) router and serves to connect the secure network 100 to devices outside the secure network. An audit server is depicted at 120. For illustrative purposes, an intruder computer system is depicted at 130.

Figure 2 is a block diagram illustrating an exemplary computer system, such as the audit server 120 depicted in Figure 1, usable on the internal secure network 100. The present invention is usable with currently available personal computers, mini-mainframes, mainframes and the like. Although audit server 120 is depicted in Figure 1 as a network device which is part of a wired local network, the audit server 120 is also envisioned as being connected to the network 100 by a wireless link.

Audit server 120 includes a bus 202 or other communication mechanism for communicating information, and a processor 204 coupled with the bus 202 for processing information. Audit server 120 also includes a main memory 206, such as a random access memory (RAM) or other dynamic storage device, coupled to the bus 202 for storing information and instructions to be executed by processor 204. Main memory 206 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by

IPR2024-00027
CrowdStrike Exhibit 1002 Page 795

processor 204. Audit server 120 further includes a read only memory (ROM) 208 or other static storage device coupled to the bus 202 for storing static information and instructions for the processor 204. A storage device 210, such as a magnetic disk or optical disk, is provided and coupled to the bus 202 for storing information and instructions.

Audit server 120 may be coupled via the bus 202 to a display 212, such as a cathode ray tube (CRT) or a flat panel display, for displaying information to a computer user. An input device 214, including alphanumeric and other keys, is coupled to the bus 202 for communicating information and command selections to the processor 204. Another type of user input device is cursor control 216, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 204 and for controlling cursor movement on the display 212. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y) allowing the device to specify positions in a plane.

The processor 204 can execute sequences of instructions contained in the main memory 206. Such instructions may be read into main memory 206 from another computer-readable medium, such as storage device 210. However, the computer-readable medium is not limited to devices such as storage device 210. For example, the computer-readable medium may include a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave embodied in an electrical, electromagnetic, infrared, or optical signal, or any other medium from which a computer can read. Execution of the sequences of instructions contained in the main memory 206 causes the processor 204 to perform the process steps described below. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 796

invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

Audit server 120 also includes a communication interface 218 coupled to the bus 202. Communication interface 218 provides a two-way data communication as is known. For example, communication interface 218 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 218 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. In the preferred embodiment the communication interface 218 is coupled to the network cable 102. Wireless links may also be implemented. In any such implementation, communication interface 218 sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information. Of particular note, the communications through interface 218 permits the transmission or receipt of audit trail information.

To assist in fully describing the present embodiment of audit reduction system the following terms are used with the following definitions. Note, however, that although a term may be herein defined, this does not necessarily exclude an established definition for the term if using the established definition is consistent with the purpose and scope of the present invention. "Audit trail information" includes (1) system audit trail records; (2) processing system log file data; and (3) processing system-maintained security state data. The audit trail information is received as native data and recorded "Authentication" entails associating a user with a system identifier. A "subject" is an active entity, generally in the form of a person, process, or device, that causes information to flow among objects or changes in the processing system state. An "object" is a passive entity that contains or receives information. Access to an object implies access to the information it contains.

An "audit agent" is a group of processes that handles the conversion of native audit files to XHAC data and transfers that data to the audit server 120.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 797

"Audit processing" is the ability to read an input stream of data and to identify what type of data it is (i.e., BSM, C2, etc.) so that transformation may occur if needed. This is the input function of the agent. "Transformation" or "conversion" refers to the changing of data from one format (i.e., BSM) to another format (e.g., XHAC). This is the output function of the agent. The communication processing portion of the agent handles all of the communication logic between the agent and the manager as well as communication between the various processes that make up the agent.

With the above-definitions, the present embodiment may be understood as an auditing system, generally indicated at 300 in Figure 3. The auditing system 300 includes the previously mentioned audit agent, indicated at 310, which resides on the audit server 120 (not shown in Figure 3). The audit agent 310 receives native audits from each of the nodes on the network 104, 106, 108, 110, 112, 114, and 118. These audit trails include, for example, 1) system audit trails; 2) system log file data; and 3) data from third party applications and programs. The data from third party applications may use their own form of logging. As is known, the third party applications are used on many different types of processing systems. All of the information from these native audits is typically transient information which is subject to change based on the current activities on the processing system. As explained in greater detail below, the audit agent 310 collects the audits in the form of an audit trail, filters the audits, reduces the number of audits and normalizes some of the audits to a standard format for output to a security indications and warnings system (an intrusion and misuse detection engine).

The logical architecture in Figure 3 illustrates an audit reduction workbench 320 which can be provided for communication with the audit agent 310. The audit reduction workbench 320 can be the personal computer 112 depicted in Figure 1. The audit reduction workbench 320 can be used to build, maintain and send compiled audit reduction scripts to the audit agent 310. The audit reduction workbench 320 can be used to build audit processing rules and to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 798

build anomaly detection rules. A configuration tool 330 can be provided which can communicate with the audit agent 310. The configuration tool 330 can be the personal computer 112 in Figure 1. The configuration tool can be used to define audit agent configurations and define audit processing parameters. Workbench 320 and tool 330 can be combined to use a single computer. Configuration tool 330 sends configuration settings to an archive 340. Audit agent 310 can reside on the audit server 120 or can reside on one or more computers in the network. As described below, audit agent 310 sends native audits to an on-line audit file storage device 350 associated with audit server 120 and sends normalized audits to an audit analysis database storage device 360 associated with audit server 120. Audit agent 370 sends real time audits to a real time monitor 370 and real time alerts to a real time alert monitor 380. Real time monitor 380 can be used to view audit alerts in near real time, view alerts and messages from other tools, view reports and activate other tools. Monitors 370 and 380 can be combined into a single computer or monitor. An audit review tool 390, which can be a personal computer, receives configuration information from archive 340, on-line audit files from on-line audit file storage device 350 and normalized audit information from audit analysis database storage device 360. An ad hoc query tool 400, which can be a personal computer, receives normalized audit information from audit analysis database storage device 360. Stored queries can be defined and processed periodically. Ad hoc queries can be generated and executed immediately. An anomaly reporting tool 410 receives normalized audit information from audit analysis database storage device 360 and can forward this information to the alert monitor tool 380. The anomaly reporting tool can generate alerts in near real time, replay audit scenarios and produce summary reports and statistics. An administrative tool 420 is associated with audit server 120 and may be in the form of a personal computer. The administrative tool 420 is used to define and maintain user descriptions/privileges, define and maintain group descriptions and define and maintain machine descriptions.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 799

As depicted in Figure 4, the logical architecture of the audit agent 310 is depicted. Two mechanisms can be used to provide audit reduction code to the audit agent 310. The first mechanism is that the code can be specified in a script 450. The second mechanism is an audit reduction compiler 460. Using either mechanism, instructions are inserted into address space within the audit agent 310. There are five matrices that include information for the audit agent to use in dealing with incoming information. The first matrix is an environment parameters matrix 500 which provides information such as where to retrieve files, where the files are located in the directory structure, the naming convention, which operating system generated the files, and so forth. The second matrix is an identification matrix 510 which identifies each of the records in an incoming data stream based on information contained within each record. The third matrix is a transform matrix 520 which, once a record has been identified, can take one of two actions. The first action is direct transform meaning that the record is written into a different format and sent to the audit analysis database storage device 360. The second action is to send the record to a reduction matrix 540 (fifth matrix) and reduce out the record. Advantageously, in the present invention, about ninety percent of the records are reduced out and only about ten percent of the records are transformed and further analysis is performed. The fourth matrix is a token matrix 530 which allows the audit agent 310 to find specific information within a record.

As previously mentioned, the audit agent 310 receives three types of native audits: 1) system audit trails 550; 2) system log file data 560; and 3) data from third party applications and programs 570. The audit agent 310 can access shared libraries 340. Normalized audits and native audits are sent by the audit agent 310 to storage devices 360, 350, respectively using a transport protocol. Status and control are also transmitted and received within audit reduction system 300 using a transport protocol.

The native audits are special purpose records that include information about system activity and transactions. These native audits include data written in

IPR2024-00027
CrowdStrike Exhibit 1002 Page 800

a particular format. This formatted information may be called an audit record. Audit trails contain records showing who has accessed a computer system and what operations he or she has performed during a given period of time. Audit trails are useful both for maintaining security and recovering lost transactions. A plurality of records constitute a file. Data and records from on-line audit file storage device 350 and archive 340 can be used for audit system 300 to reconstitute actual events that occur within the processing system.

Refer now to Figure 5 where a native audit record is illustrated. The native formats can be in one of two formats; either positionally based or contextually based. A contextually based record is, for example, two different business cards from two different organizations where the information is positioned in different locations on the business card. On one business card the name may be in the middle of the card and on another business card, the name may be in the upper left hand corner. The context of the information, not the position, controls in a contextually based record. The best example of a contextually based format is syslog where the data can be free form. In this format the next field is determined by the previous field.

Syslog Example (Each field determines what should follow by the value in the field.)

1. VMUNIX: Reboot: This is a Reboot Message.
2. SendMail: Error reading SendMail config.
3. This is a plain text string.

For simplicity of explanation, the present invention is described in relation to the positionally based record depicted in Figure 5. A positionally based record includes information contained in particular positions in the record. The audit agent 310 has access to templates to recognize contextually based records and positionally based records and obtain information from either type of record. However, for ease of discussion, the processing of contextually based records is not further discussed herein. It should be appreciated that the record illustrated in Figure 5 is simplistic and is only being used for ease of explanation. It should be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 801

further appreciated that many other positionally based records and contextually based records can be used in the invention.

A brief overview of how the audit agent 310 uses the information contained in an audit trail record is now provided with a more detailed explanation provided in the detailed flow diagrams (Figures 8-39) and discussion thereof. A record is a stream of binary information. The record illustrated in Figure 5 has an origin O which is the beginning of the record. The record comprises a plurality of fields. The first field (Field A) contains information concerning record length (RL). The second field (Field B) contains a primary discriminator (PID), for example 6, in the second field. The primary discriminator includes information on the type of record. There may be additional fields here, in this example, designated C, D, E, and perhaps other data thereafter.

For purposes of explanation, the previously described record having a PID of 4 has a format identifier (ID). For example, this record format may have an ID = 5. All the records that are received having an ID = 5 are going to be processed using the same set of instructions. The record is compared against a global format (see Figure 5B) for an ID = 5. An immediate warning or alert message can be sent to a system administrator if certain criteria are met. The criteria might be that someone is trying to hack into the data stream. For example, if invalid data is in the incoming record stream, it may be an indication that someone is hacking into the data stream. Expert systems use this method for determining intrusions and misuses. The global format contains a set of instructions for all formats, including format 5 This way cross format rules can be supported. The global format (GF) includes information as to which output format (OF) that is associated with this input record. The output format can be an ASCII representation of the input record, or it can be a normalized record type, or it can be any desired output format.

After the record has been compared against the global format and an output format determined, then the record is compared against a set of tables called record entity tables (see Figure 6). As depicted in Figure 6, global items, such as

IPR2024-00027
CrowdStrike Exhibit 1002 Page 802

format, record entities (control REC), and the like are always operated on first to provide for cross format operations. There are specific formats, for example, format 5 may be for system logins (syslog). Another format might be format 19 for SOLBSM. Another format is format 21 for SLOLMW. From the global format at 610, the process continues to 620 in which each record entity is provided with a number. A matrix is provided at 630 for conversion rows. For example, REC record size, record type, date and user, and so forth. A matrix at 640 includes templates for matching records for logins, logouts and mounts. A matrix is provided at 650 which includes instructions for user equal route, record type equal login and date greater than a certain date such as October. A scenario matrix is provided at 660 which can be a chronological matrix, a bad logins matrix and a bad super-user scenario. At 670, a template matrix is provided which might include a login template, bad login template and a bad super-user template. At 680, a build row template is provided which might include a reduced PID build row template, reduce user build row template and a reduce record-type build row template. At 690, an action template is provided which could either be reduce, transform or reduce. At 700, various scenarios are created as instances or "copies" of the scenarios already activated. There are various scenarios that might be currently in progress such as a three bad logins scenario, three bad logins on JHW's computer, three bad logins on WAC's computer or a bad super-user (SU). There are various pointers denoted by dashed lines which are used based on the scenarios for the template matrix 670, the build row template 680, and the action template 690. If a reduction is currently in progress at 692, there is an elimination list of matrix. These lists where entries are created "on the fly" by successfully completed scenarios by the build rows. At 694, there is an elimination matrix which includes certain elements which were previously audit records.

The audit agent 310 starts going through these record entities tables and tries to identify the record typically based on the primary discriminator. For example, in this simplistic view, if the primary discriminator includes a 6 (see

IPR2024-00027
CrowdStrike Exhibit 1002 Page 803

Figure 5A), then a record entity corresponding to a record type = 6 is activated. This identifies the record. There is an action associated with the identification of the record.

The action illustrated in Figure 6 is a decision whether to reduce the record (throw the record out) from further consideration or normalize the record into a standardized format and then to save the record for further analysis by the intrusion and misuse detection engine. There are three types of reductions: simple-simple, simple-complex, and complex-complex. Simple-simple means one record is compared and interpreted against one template. Simple-complex means one record is compared and interpreted against many templates, but it takes only one record to move from a comparison phase state to an interpretation phase. Complex-complex means many records are compared and interpreted against many templates.

In a simple-simple reduction, the record is reviewed against a single record entity or template, and either there is a conversion, or the record is reduced out. If reduced, the record is not converted to another format but is sent to the on-line audit file storage device 350. Then the next record is analyzed.

Reduction occurs in two phases: comparison (to see if something has happened) and interpretation (do something about it). To connect records that are somehow associated, the software uses defined fields as linkage fields. Linkage fields can include the process identifier (PID) (linkage or grouping by process), user identifier (UID) (linkage by user), etc. For example, the information contained in field C could be the linkage field. All record-type = 6 records are operated on through the linkage field. In this example, all the record-type = 6 records that have a matching port ID or processing ID are matched for that record and then all the subsequent records that match that record are reduced out. For example, all records having a record-type = 6 in field B and an 8 in field C (8 representing the matching part or processing ID).

In a complex-complex reduction, one record entity might find a match for a record-type = 6 and if field C contains an 8, there is a whole series of records to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 804

match. In this example, the audit agent may search for another record that has a record-type = 6, and so forth. Additional pattern matching could be required.

A simple-complex reduction is depicted in Figure 7B in which a record-type = 6 in field B and a PID of 1234 in field E is compared against a template in the comparison phase. In the interpretation phase all records with a PID of 1234 are compared against an elimination list.

A complex-complex reduction is depicted in Figure 7C in which once a record having a record-type = 6 in field B and a PID=1234 in field E is located in the comparison phase, all subsequent records having record-types equal to either 6 or 8 are compared against all records with a PID=1234 during the interpretation phase.

In Figure 7D, a flow diagram of the steps of Figure 6 is illustrated. In step 610, incoming audit records are received. At step 620, an audit record is in the view and each of the audit records is matched against templates and elimination lists to determine if each audit record is to be reduced or further processed. If the audit record is marked for reduction at step 630 then, at step 640, the record is copied for an in progress active scenario (in case scenario fails) as discussed in detail below. At step 650, the audit record is either reduced or sent for conversion.

Actions can be taken depending on the matching based on the fact that the audit agent 310 failed or succeeded in finding a match. For example, a match may be expected such as in a financial transaction where a certificate is provided. If a certificate is provided, the records should be in a certain order or the fields in certain records should contain certain information, then the system may generate an alert that this certificate is not adhering to the standard methodology for a certificate. Another example is logins and logouts. For example, if record-type = 6 in one record and then record-type = 6 in another record for one computer on the network, this means this is a login (record-type = 4) and this is a logout (record-type = 6) and there is over twenty-four hours in between these two records, the audit agent 310 an alert may be provided because one machine can

IPR2024-00027
CrowdStrike Exhibit 1002 Page 805

not be logged on for twenty-four hours, so that means that the user left their machine logged on. This is a potential security breach because a misuser could be using the logged on computer. In a complex-complex situation, a record may be matched against hundreds and hundreds of templates and different fields from each record compared against different templates. One match may cause additional matching operations to be performed.

An example of a simple-complex reduction includes successful logins on a SUN Solaris 2.5 system. With all audits enabled, the system will generate about 150 audit events – all considered normal activity. Using the simple-complex methodology, a scenario would become active at the receipt of a record with a record-type of 4 (Example) subsequently an elimination list entry is build to reduce out the 149 subsequent audits based on the PID linkage field. The initial record generates the login output record. An example of a complex-complex scenario includes the tracking of user operations to ensure that the sequence is completed correctly. In this application of the technology it is necessary to step through the sequence of records while in the comparison phase to identify if the user actions occurred correctly. Some records are necessary to the completion of the scenario but some intermixed (within the linkage constraints) records may be ignored. All comparisons in the template list must succeed prior to switching to the interpretive phase.

Specifically, the analysis of a chronology (cron) job scenario requires a complex-complex definition. First, the system must detect a login event. Then a number of activities occur associated with the system starting the job, i.e., the generation of a shell, the reading of the even tab file.

In summary, the audit agent 310 performs a first line analysis and reduces the audit records to be output and then outputs the audit records. The audit agent can also wait for additional records. The analyzed, reduced and transformed data is then output to audit analysis database storage device 360. This data can then be further analyzed by the intrusion and misuse detection engine disclosed in a patent application entitled "Method and System for Detecting Intrusion into and Misuse

IPR2024-00027
CrowdStrike Exhibit 1002 Page 806

of a Data Processing System" assigned to the instant assignee and filed on even date herewith and incorporated by reference into this specification in its entirety.

With this brief overview in mind, the details of the present invention can be understood with reference to Figure 8 where a top-level flowchart is depicted according to the present invention. The flowcharts illustrated in Figures 9-39 are details of the steps in Figure 8. At step 800 in Figure 8 the process is started. At step 805, the process is initialized. The initializing step is illustrated in greater detail in Figure 9. At step 910, in Figure 9, tables are built, by setting up, for each format (positional and contextual), the five matrices 500, 510, 520, 530, 540 discussed above with respect to Figure 4. At step 920, communications are set up by initializing any sockets, etc. All the matrices are populated within the address space, so now the instructions are available for the audit agent 310 to start processing. At step 930, the initializing step 805 is complete and the process proceeds to the sets up audit processing step 810 in Figure 8.

The set up audit processing step 810 is illustrated in detail in Figure 10 and at step 1000 the process is started. At step 1005, the audit agent 310 decides whether to wait for additional format types. For example, there may be one or more audit types that are positionally based, and one or more audit types that are contextually based. If there are no more types to wait for, at step 1010, the set up audit processing step 810 is complete. At step 1015, the audit agent parses an intermediate audit reduction (AR) file, and builds tables. A file is created that allows the audit agent 310 to make temporary changes to the file without making permanent changes. At step 1020, the configuration of the record is read and the type of configuration is determined, based on the environment constraints 500. The environment constraints 500 is used to determine where the record comes from. At step 1025, a parse reduction file can be set up although it is not necessary. At step 1030, a scan directory step provides information regarding the location of the audit files. For example, the audit files may be sent by the audit agent 310 in real time or may be extracted from the on-line audit files database 350. At step 1035, it is determined whether any audit files exist. If there are audit

IPR2024-00027
CrowdStrike Exhibit 1002 Page 807

files, then at step 1040, the file table is updated. Otherwise at step 1045 the system returns to step 1005. When all of the types of records have been set up, the set up processing step is completed at step 1010. This set up audit processing step 810 is performed for all the different format types that are going to be processed.

Returning to Figure 8 at step 815 , there is check for a terminate signal from the server or alert monitor tool 380 to shut down. At step 820, there is check for any change signals to change the current configuration. A change signal would be sent from the anomaly reporting tool 410. At step 825, if there is a change request, changes are made to the tables and so forth.

At step 835, the process audit types step is begun. Step 835 is conducted sequentially for each audit type. The process audit types step 835 is illustrated in detail in Figure 11 and at step 1100 the process is started. At step 1105, the audit type is checked. At step 1110, if there are no files to process then the process audit types step is complete. At step 1115, it is determined whether there are any files to process. At step 1120, if there are files to process then the files are processed and after the files are processed the process continues to step 1135. At step 1125, audit files are selected based on criteria. The criteria includes a time range as well as a naming convention for the current source. At step 1130, once files have been selected the process returns to step 1115. If there are no new files, then the process proceeds to step 1135. Step 1135 returns the process to step 1105.

The process file step 1120 is illustrated in greater detail in Figure 12. At step 1200, the process file step 1120 is started. At step 1205, the programmer use file location is checked against a type equals file location to see if this is an open file. At step 1210, the file is checked to see if has been previously processed, and if any new data has been added to this file. At step 1215, the amount of process time for the file is checked and if it exceeds a predetermined amount of time, then at step 1220, the process is completed. At step 1225, the file location is checked against file size. If the file location equals file size no new data has been added to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 808

that particular record and the process proceeds to step 1230. If the file location does not equal the file size, then at step 1235, this means that the operating system has only written a partial record. At step 1240, if the file size minus the file location size is less than a buffer size then additional records can be added to the buffer until, at step 1245, there is a full buffer. The buffer includes a plurality of records forming a portion of a file. At step 1250, the file buffer size is read and then at step 1255 the process buffer step 1255 is started. Step 1255 loops back to step 1230 and then to step 1215 until all of the records have been read.

The process buffer step 1255 is illustrated in greater detail in Figure 13. At step 1300 the process buffer step is started. At step 1305, a pointer is set to the beginning of the buffer. The pointer advances as it goes through the buffer and identifies records. At step 1310 the process waits for more records. If there are no more records, at step 1315 the file location is equal to buffer location pointer. At step 1320 the file table is updated and at step 1325 the process waits until the next cycle.

At step 1330, if there are additional records, the additional records are identified and the record size determined. At step 1335, if there are no more records (end of the audit file) the process proceeds on to step 1360 where the file is marked, and the buffer location pointer is set to zero at step 1365. If it is not the end of the audit file, at step 1335, the process checks for a file header at step 1340. If there is no file header, at step 1340, it must be determined, at step 1345, whether to perform further processing, meaning whether to convert the record to a standardized format or to reduce the record out by advancing the buffer at step 1355. At step 1350, the conversion is performed which is discussed in greater detail below.

Figure 14 is the locate event type flow chart. At step 1400 the process is started. At step 1410 the primary discriminator is located. At step 1420 the ID pointer is set to a primary discriminator offset. The primary discriminator (PD) performs the grossest level of filtering. It provides for a rapid narrowing of potential rules for the process to check against. Similar in concept to a sort, i.e., if the PD is "X" and only four rules apply to a record with a PD of "X" the processing can be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 809

saved of rules for PDs of "Y", "Z", and "A" because they can't apply. The PD offset tells the process where to find the PD within the record (it varies depending on the format, and in certain cases doesn't exist). At step 1430, the match function is performed and at step 1440, the process is complete.

The match function step 1430 is illustrated in greater detail in Figure 15. At step 1500 the match function step is started. At step 1505, the match index is set to zero meaning that the process starts with the first match record. Records that are matched need to be checked again for additional matches. At step 1505, the process is set to the beginning of a match table. At step 1510, the process looks for the end of the match table and checks for the end of file. As previously mentioned above with respect to step 1410, the primary discriminator has already been identified. The match function is a table in which is stored the particular templates that a record should be compared against based on the primary discriminator. This is the same as the records entity templates of Figure 5). Once the process reaches the end of file, the process exits at step 1515. If the process is not at the end of the match table, then processing is continued at step 1518. At step 1520, there is a binary comparison of two memory locations to determine if there is a match between the two values. This can be two values in two fields or one value in the record and the other value in the template. If there is not a match at step 1520, then the pointer is advanced in the match table and the process goes on to the next match record at step 1565. At step 1520, if there is a match, at step 1525 the match command is set to zero. Match commands are the "holes" in the templates, i.e., how to conduct the match. A match command table is another table that is a sub grouping of instructions from the original match table. At step 1530, using the match command table, it is determined if the pointer is at the end of the table. If there are additional templates to compare then the process proceeds to step 1535. If the match is equal to zero at step 1560 then there are no more matches. There was at least previous one match for the primary discriminator but not using any of the additional templates. If there are additional matches, at step 1535 then the process proceeds to step 1540, and more match commands are provided if there is a positive match result at step 1545. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 810

match result is produced at step 1545 and at step 1550 the pointer is set to continual and then the process exits out at step 1530. If the process is not at the end of the match command, at step 1530, the absolute value of the match is taken which is always expected to be positive if it is successful. Commands can be successful if the command fails. The audit reduction agent 310 requires a positive return from each command processed, otherwise the match fails. However, in certain cases, negative logic is used (i.e., if something is not equal or fails to match). The expected state is present so that when a failure is expected and one is obtained, this is considered success. The match command is incremented at step 1560 and this loop is continued until all the match commands for this match record have been checked. Depending on the match obtained at step 1560, then at step 1565 it is determined whether to continue with the additional match records or exit.

The process command step 1535 is illustrated in greater detail in Figure 16. At step 1600 the process command step is started. At step, 1605 an arbitrary command is checked to see if it is greater than 95. If the command is greater than 95, then it is considered conditional. The process is checking here to see if there is an exit or fault command. At step 1610, the process checks to see if the command is less than 100 and an X flag is set at step 1615. At step 1620 there are cases when it is desired that the result obtained returns false. At steps 1605-1615, if the command is greater than 95, the result is expected to be false. So a true is returned in that case. Processing can not continue unless a true is returned. These arbitrary values have to pre-set so that the process is successful in recognizing a false state. At step 1620 the case instructs the audit agent 310 to operate on any particular field within the data. This moves the pointer in the data to a new location. At step 1625, the match pointer is set equal to the buffer location pointer plus an offset to perform a comparison. At step 1635 a binary comparison occurs with the pointer being in the table. If the X flag is set in step 1615, then the X flag says that the process is supposed exit if this is true. A primary discriminator has been identified so the process has the functionality to create a program about how to continue to identify this record. The record or data can be compared to other data, pointers can be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 811

advanced to look for other values within the record and the process command takes what is written in the command table and executes the commands in the command table.

Step 1625 provides the ability to advance the pointer within the data and look at other fields besides the primary discriminator. The binary compare step 1635 can be used to compare a field of data against other fields of data in other databases. For example, the data could be contained within other tables, memory or even contained in interface 118. Step 1640 is a determination to set the X flag based on a set of conditions. The audit education agent 310 requires a positive return from each command processed, otherwise the match fails. However, in certain cases, negative logic is used (i.e., if something is not equal or fails to match). The expected state is present so that when a failure is expected, and one is obtained, this is considered success (X = exit on true). If certain conditions are met at step 1645, the process exits. If the conditions are not met at step 1650, a retrieval is obtained which equals the results plus a command which is a numeric value (command value numeric). At step 1655, the process exits. Step 1630 is an error handler for an unknown value.

The perform conversion step 1350 in Figure 13 is illustrated in greater detail in Figure 17. At step 1700, the process is started and at step 1705, the conversion index is set to 0. At step 1710, conversion is performed until the end of table is reached or until a match is found. If no match is found, then at step 1715, the native audit is reduced. At step 1720, it is determined whether a reduction has been performed. If no reduction has been performed at step 1725, the record is written to native audit on-line audit file 350. If the reduction has occurred at step 1720, then the process is complete at step 1730. At step 1710, if the end of file has not been reached, at step 1735, a conversion row is started. There are 1 to N number of conversion rows. There is an input conversion row and an output conversion row which tells the audit agent 310 how to convert the data and where to put the converted data. At step 1740, if the length of the converted row is equal to –1, then the audit agent 310 does not know what the length of the converted record is. At

IPR2024-00027
CrowdStrike Exhibit 1002 Page 812

step 1745, a delimiter is determined for that record. If the conversion row is equal to −1, then at step 1755, the token entry is located. The token entry delimits the field definition. For example, it might be a colon, or some other type of data that provides information that the end of field has been reached. If there is no token as determined at step 1760, then at step 1765 and 1770, the length of the audit record is determined. At step 1775, if the token is null, then the string length is determined. Using either step 1770 or 1775, the string length can be determined and the amount of data limited. In either event, at step 1750, other codes are executed to perform specific conversion on the data. For example, there is a registered call back function which gets executed because the data is being converted from an input format to an output format. The process loops back to step 1710 until all the conversion rows and data have been converted. All the data will be mapped into the conversion table to be converted.

Reduction step 1715 is illustrated in greater detail in Figure 18. At step 1800, the reduction step is started. At step 1805 there is a determination step to determine whether the record is in an elimination list 1810 by comparing the record against the elimination list 1810. The record is matched against every entry in the elimination list 1810. If the record matches any of the entries, the record will be reduced out. At the beginning, there are no active scenarios in the elimination list 1810. There are three lists of scenarios. The first is the list of possible scenarios that can happen as defined in the memory by the AR programmer. The second is a list of active scenarios in memory. This could include scenarios; for example, SC1-SC6, as depicted below. Because certain events have occurred in the audit stream (chronological audit data in order of generation), a scenario is said to be "in progress" or "active". This could include scenarios SC3, SC3, SC1, and SC6 as depicted below. This means the audit agent 310 will look for subsequent audit records (as matched against templates) as opposed to the already reviewed audit records.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 813

Life cycle of a scenario:

Can happen	Are happening	Have happened
Possible Scenarios (as defined by the ARL Programmer	In Progress Scenarios (Comparison Stage) Have Been Activated by Events	Elimination List Generated by Successful Scenarios to Reduce Out Additional Records
SC1	SC3	SC3 PID=21
SC2	SC3	SC3 PID=22
SC3	SC1	SC3 PID=23
SC4	SC6	SC3A PID=100
SC5		SC3APID=101
SC6		SC3APID=102
Etc.		SC1 Etc.

At step 1815, there is a determination as to whether there are any currently active scenarios. A scenario includes two or more identifiers within a record. When the audit agent first begins, there are no active scenarios in the elimination list but scenarios will be added as the process continues as discussed in detailed below. At step 1815, it is determined whether there are any active scenarios. An active scenario list is provided at 1835. How the active scenario list is created is discussed below. For the purposes of Figure 18, it must be assumed that there are some active scenarios in list 1835. Step 1830 determines whether the record matches any current scenarios in the active scenario list. If there are no matches, then at step 1840, it should be determined whether to create new scenario parameters. This means that a new scenario is being added to the active scenario list 1835. If step 1830 is positive and matches located, then the record is compared against other templates. The active scenario list 1835 means that the scenarios are currently waiting for a record. For example, there may be thirty-two active scenarios, but the active scenario is waiting for a particular record. At step 1825, the reduction process is stopped.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 814

The record in elimination list step 1805 is illustrated in greater detail in Figure 19. At step 1900, the process is started. At step 1905, the reduction list is selected to search by format. At step 1910, it is determined whether a match has occurred against a list of reduction entries. If no match has occurred, then the process is completed at step 1915. If the process is continuing to perform matches, then at step 1920, timers are checked. At step 1925, the ordinality is validated. Most comparisons and templates work on ordinality of "fields" (areas of data within a record). Step 1925 ensures pointers are at the appropriate location based on ordinality. At step 1930, the audit records are checked for completeness. At step 1935, it is determined whether the audit data matches the reduction list data. If the audit data does not match the reduction list data, then the process returns to step 1910. If the audit data does match the reduction list data, then at 1940, the reduction list match is processed.

The processed reduction list match step 1940 is illustrated in greater detail in Figure 20. At step 2000, the process is started. At step 2005, it is determined whether the reduction sequence could be decremented. A grouping of elimination list records which are associated – decrementing refers to the popping of a elim list record from the associated record "stack", i.e., making the next record in the sequence active. If the reduction sequence can be decremented, then at step 2010, the reduction sequence is decremented. If the reduction sequence can not be decremented, then at step 2015, it is determined whether the matched reduction entries life span is temporary. If the matched reduction entries life span is temporary, then at step 2020, the matched list entry is removed. At step 2025, the process is complete.

Determination step 2005 is illustrated in greater detail in Figure 21. At step 2100, the process is started. At step 2105, the previous list entry is compared against criteria. Criteria can be any values, ranges, etc., based on scenario instructions. At step 2110, the next list entry is compared against criteria. At step 2115, it must be determined whether the previous and next records meet the criteria. If the previous and next records meet the criteria, then at step 2120, the sequence may be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 815

decremented. And at step 2125, the process is complete. If the previous and next records do not meet the criteria, then at step 2130, the sequence may not be decremented and process is completed at step 2125.

The compare previous list entry against criteria step 2105 is illustrated in greater detail in Figure 22. At step 2200, the process is started. At step 2205, it is determined whether the previous record is null. If the previous record is null, then at step 2210, the previous record meets the criteria. If the previous record was not null, then at step 2215, it must be determined whether the matched IDs are equal. If the matched IDs are not equal, then the previous record meets the criteria at step 2210. If the previous record and/or row, then the previous record meets the criteria at step 2210. If the previous record and/or row is not determined, then at step 2225, it must be determined whether the previous record has a sequence of 1. If the previous record does not have a sequence of 1, then the previous record meets the criteria at step 2210. If the previous record does have a sequence of 1, then at step 2230, the previous record does not meet the criteria. From either step 2210 or 2230, the process proceeds to step 2235, where this portion of the process is complete.

The compare next list entry against criteria step 2110 is illustrated in greater detail in Figure 23. At step 2305, the process is started. At step 2310, it is determined whether the next record is null. If the next record is null, then at step 2315, the next record meets the criteria. If the next record is not null, it must be determined at step 2320, whether the matched IDs are not equal. If the matched IDs are not equal, then proceed to step 2315. If the match IDs are not equal, then at step 2325, it must be determined, is it the next record and/or row. If it is, then proceed to step 2315. If it is not, then step 2330 must be determined as the next record life span temporary. If it is, then proceed to step 2315. If it is not, then at step 2335, it must be determined whether the next record sequence is greater than 1. If it is, then proceed to step 2315. If it is not greater than 1, then at step 2340, it must be determined whether the next record sequence is equal to 1. If it is not, then the next record meets the criteria at step 2315. If it is, then at step 2345, the next record does

IPR2024-00027
CrowdStrike Exhibit 1002 Page 816

not meet the criteria. From either step 2315 or 2345, the process proceeds to step 2350 where the process is complete.

The decrement reduction sequence step 2120 is illustrated in greater detail in Figure 24. At step 2400, the process is started. At step 2405, it must be determined whether there are entries in the list and the matched IDs that are equal. If there are none, then the process proceeds to step 2410 where the process is complete. If there are entries in the list and match IDs that are equal, then at step 2415 it must be determined whether the sequence is greater than 1. If the sequence is not greater than 1 then the process returns to step 2405. If the sequence is greater than 1, then at step 2420, the entries sequence number is decremented by 1. Then the process returns to step 2405.

Steps 1845 and 1855 both proceed to step 2500 as illustrated in Figure 25. Figure 25 depicts the process of a complex second stage interpretation. At step 2505, the next template record is read in. At step 2510, it is determined whether the end of the template list has been reached. At step 2510, if the end of the template list has not been reached, then it is decided whether to add this elimination scenario (plurality of templates) to the elimination list 1810. If the answer to step 2515 is yes, then at step 2520, the plurality of templates which make up a scenario are added to the elimination list. If the answer to step 2515 is no, then the process stops at step 2525. At step 2530, because the scenario did not generate any matching templates, the scenario is removed at step 2530. At step 2535, the scenario is added to a post-mortem list thereby taking the scenario off of the active scenario list 1835. At step 2540, it is determined whether there are more scenarios to consider. If there are more scenarios to consider, the process proceeds to step 2525 and stops. If there are no more active scenarios, then at step 2545, the process continues to step 2525 and the process is stopped.

The removed scenario step 2530 is illustrated in greater detail in Figure 26. At step 2600, the removed scenario step is started. At step 2605, the counter associated with the open scenario list is adjusted by one. The scenario is deleted

IPR2024-00027
CrowdStrike Exhibit 1002 Page 817

from the active scenario list 1835. At step 2615, the open scenario list is updated and at step 2620, the removed scenario step is stopped.

The add to elimination list step 2520 is illustrated in greater detail in Figure 27. At step 2700, the add to elimination list step is started. At step 2705, the scenario is added to the elimination list 1810. At step 2715, the open elimination list counter is incremented. At step 2720, it is determined whether the open elimination list has reached a maximum size and used up all the memory. If the elimination list 1810 has reached a maximum size, then a scenario is deleted at step 2730 and the open elimination list is revised at step 2735. From either step 2720 or 2735, the process is then stopped at step 2725.

The add scenario step 2705 is illustrated in greater detail in Figure 28. At step 2800, the process is started. At step 2805, the audit agent 310 gets the action (see Figure 8). At step 2810, the simple-complex flag is retrieved. At step 2815, it must be determined whether there are still elements in the scenario list. If there are not, then the process proceeds to step 2820 and is complete. If there are still elements in the scenario list then the action is processed at step 2825.

The action step 2825 is illustrated in greater detail in Figure 29. At step 2900, the process is started. At step 2905, template information is retrieved. At step 2910, it must be determined whether the action is to reduce. If the action is to reduce, then at step 2915, a new reduction list entry is set up. If the action is not to reduce, then at step 2920 it must be determined whether there are any more build rows for this template. If there are not, then at step 2925, the process is complete. If there are more build rows for this template, then at step 2930, the build row is processed. At step 2935, it must be determined whether the action is to reduce. If the action is to reduce, then a new reduction list entry is set up. If the action is not to reduce, then the process returns to step 2920.

The set up new reduction list entry step 2940 is illustrated in greater detail in Figure 30. At step 3000, the process is started. At step 3010, space is allocated for a new reduction entry. At step 3020, the interpretation time is set to live. At step 3030, the match _ID is set and at 3040, the process is complete.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 818

The set interpretation time to live (ittl), step 3020, is illustrated in greater detail in Figure 31. At step 3100, the process is started. At step 3105, it is determined whether this is a new scenario. If it is not a new scenario, then at step 3110, the entries time to live is listed as equal to the previous list entries ittl. At step 3115, the process is complete. At step 3105, if this was a new scenario, then at step 3120, it must be determined whether the scenarios ittl is valid. If the scenarios ittl is valid, then at step 3130, the entries time to live is listed as equal to the scenarios ittl. If at step 3120, the scenarios ittl is not valid, then at step 3125, the entries time to live is listed as equal to 30 seconds. From either steps 3125 or 3130 the process is complete at step 3115.

The build row step 2930 is illustrated in greater detail in Figure 32. At step 3200, the process is started. At step 3205, the starting instructions and number of instructions are retrieved. At step 3210, the build rows instructions are processed. At step 3215, it is determined whether the process instructions were successful. If the process instructions were not successful, then a return value is listed as false at step 3220. If the process instructions step 3215 was successful, then at step 3225, it is determined what was the action to reduce. If the action was to reduce at step 3225, then at step 3230, the reduction parameters are checked. At step 3235, the entry is added to the reduction list. At step 3240, the process is complete. At step 3225, if the action was not to reduce, then at step 3245, it must be determined whether the advance record flag was set. If the advance record flag was not set, then the process is completed at step 3240. If the advance record flag was set, then at step 3250, the new output record is posted. At step 3255, the output record is free and at step 3240, the process is complete.

The process to build rows instruction step 3210 is illustrated in greater detail in Figure 33. At step 3300, the process is started. At step 3305, spaces are allocated in the memory for the primary identifier (PID). At step 3310, the list entries sequence field is set. At step 3315, the list entries ordinality field is set. At step 3320, a pointer is positioned to the primary identifier in the audit record. At step 3325, the primary identifier is stored from the audit record to the list

IPR2024-00027
CrowdStrike Exhibit 1002 Page 819

entries. At step 3330, the list entries primary discriminator length field is set. At step 3335, the data type of the primary discriminator and list entry is set. At step 3340, the list entries operations field is set. At step 3345, the list entries life span field is set. At step 3350 the process is complete.

The active scenario list step 1845, is illustrated in greater detail in Figure 34. At step 3400, the process is started. At step 3405, an element is added to the active scenario list 1835. At step 3415, open scenarios are added. At step 3420, the process is complete.

The reread parameters step 830 is illustrated in greater detail in Figure 35. At step 3500, the process is started. At step 3505, the process waits for more types of audit data. If more data types are received, at step 3510, there is a change the audit reduction file. At step 3515, the intermediate audit reduction file and tables are parsed. If there is no change in the audit reduction at step 3510, then the configuration is changed at step 3520. If the configuration is changed at step 3520, then at step 3525, the configuration type is read. From either step 3515 or 3525, the process continues to step 3540 and then loops back to step 3505. When all of the audit types have been completed, then the process is complete at step 3545.

The package audit data step 840 is illustrated in greater detail in Figure 36. At step 3600, the process is started. At step 3605, the packet built is equal no. At step 3610, the process waits for more types of audit data. At step 3615, if the data is being received in bulk, it must be determined what type of data is being received. If it is bulk data then it must be packetized at step 3620 and if the packet is not completed, then at step 3625, the unpacketed data is sent to a buffer and then to step 3640. After the packet is complete or if the data is not a bulk type data, then step 3630, AITP (Audit Information Transfer Protocol) packets are built. At step 3635, a signal is sent that the packet is built and the process proceeds to step 3640. After all of the data types have been received in step 3610, then the process proceeds to step 3645 where the return packet is built and at step 3650 the process is complete.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 820

The send data step 850 of Figure 8 is illustrated in greater detail in Figure 37. At step 3700, the process is started. At step 3705, a message is sent that the packet has not been sent. At step 3710, the process waits for more native audit types. At step 3715, it determines whether the data is being received in bulk. If the data is being received in bulk, then at step 3720, it determines the type and the amount of time to process the bulk data. At step 3725, it must be determined whether the time has been exceeded to process the bulk data. If the time has not been exceeded, then the process proceeds to step 3745 and back to step 3710. If the time has been exceeded at step 3725, then at step 3730, then an AITP packet. At step 3735, the time sent is reset. At step 3740, a signal is sent that the packet has been sent. The process then proceeds back to step 3710. From step 3710, the process proceeds to step 3750 where a return packet is built and at step 3755, the process is complete.

Figure 38 is directed to a configuration change for the audit reduction agent 310. At step 3800, the process is started. At step 3805, it is determined whether a configuration change command has been received. At step 3810, if a configuration change command signal has been received, then the configuration is changed at 3810. After the configuration change is posted at step 3810, the process is completed at step 3830. If there has been no configuration change at step 3805, then at step 3815, it is checked to determine whether there has been an audit reduction change. If there has been an audit reduction change then at step 3820, the audit reduction change is posted and the process is completed at step 3830. If there has been no audit reduction change at step 3815 then at step 3825 the process is restarted. At step 3835, a stat file is ridden. At step 3840, the stat file is initialized. At step 3845, processing is set up. From either steps 3825 or 3845, the process proceeds to step 3830 and is complete.

The right stat file step 3835 is illustrated in greater detail in Figure 39. At step 3900 the process is started. At step 3905, the process waits for more data types. If more data types are to be received, then at step 3910, the type of data configuration is written. At step 3915, the current AR format is written. At step

IPR2024-00027
CrowdStrike Exhibit 1002 Page 821

3920, the current AR reduction type is written and the process loops back to step 3905. If there are no more data types to be received, then at step 3925, the stat file is written and at step 3930, the process is complete.

It should now be apparent from the foregoing detailed description that a method of analyzing native audit data has been described. Advantageously, the method compares an audit record against one or more templates and against one or more audit records to determine whether the audit record might be indicative of a potential security threat. If the audit record is not considered to present a potential security threat, then the audit record is reduced. If the audit record or records represent a potential security threat, then the audit record or records are normalized to a standardized format for further evaluation by a misuse and intrusion detection engine.

It will be readily seen by one of ordinary skill in the art that the present invention fulfills all of the objects set forth above. After reading the foregoing specification, one of ordinary skill will be able to affect various changes, substitutions of equivalents and various other aspects of the invention as broadly disclosed herein. It is therefore intended that the protection granted hereon be limited only by the definition contained in the appended claims and equivalents thereof.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 822

What is Claimed Is:

1. A method of reducing the volume of native audits received from at least one operating system, each of the native audits being in a particular format, comprising:

identifying the particular format for each of the received native audits;

comparing each of the received identified native audits against at least one template and determining if each of the native audits matches at least one template; and

reducing each of the matched native audits.

2. The method of claim 1, further comprising transforming selected ones of the native audits to a standardized output.

3. The method of claim 2, further comprising outputting the transformed audits to a transformed audit storage device.

4. The method of claim 1, wherein the native audits comprise at least one of system audit trails; system log file data; and data from third party applications and programs.

5. The method of claim 1, comprising associating each of the native audits with a global format template, each of the global format templates including a listing of additional templates against which each of the associated native audits should be compared.

6. The method of claim 2, comprising associating each of the native audits with an output template for transforming the native audit to the standardized output format.

7. The method of claim 1, further comprising outputting the native audits to a native audit storage device.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 823

8. The method of claim 1, comprising comparing one of the received native audits against a single template and either reducing the native audit and sending the native audit to a native audit storage device or transforming the native audit and sending the transformed audit to the transformed audit storage device.

9. The method of claim 1, comprising comparing one of the received native audits against a plurality of templates and either reducing the native audit and sending the native audit to a native audit storage device or transforming the native audit and sending the transformed audit to the transformed audit storage device.

10. The method of claim 1, comprising comparing a plurality of the received native audits against a plurality of templates and either reducing the plurality of native audits and sending the plurality of native audits to a native audit storage device or transforming the plurality of native audits and sending the plurality of transformed audits to the transformed audit storage device.

11. The method of claim 10, wherein the native audits are in one of a contextual format or a positional format.

12. The method of claim 1, comprising setting up at least one possible scenario and comparing the received native audits against the at least one possible scenario and activating the at least one possible scenario to become an in-progress scenario.

13. The method of claim 12, comprising comparing subsequently received native audits against the in-progress scenario, a successfully completed scenario becoming an elimination list.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 824

14. The method of claim 13, comprising comparing each of the received native audits against the elimination list and if the received native audit matches a record on the elimination list, reducing the matched native audit.

15. An article, comprising:

at least one sequence of machine executable instructions;

a medium bearing the executable instructions in machine readable form;

wherein execution of the instructions by one or more processors causes the one or more processors to;

identify the particular format for each of the received native audits;

compare each of the received identified native audits against at least one template and determine if each of the native audits matches at least one template; and

reduce each of the matched native audits.

16. A computer architecture for reducing the volume of native audits received from at least one operating system, each of the native audits being of particular format, comprising:

identifying means for identifying a particular format for each of the received native audits;

comparing means for comparing each of the received identified native audits against at least one template and determining if each of the native audits matches at least one template; and

reducing means for reducing each of the native audits.

17. A computer system, comprising:

a processor; and

a memory coupled to said processor, the memory having stored therein sequences of instructions, which, when executed by said processor, causes the processor to perform the steps of:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 825

identify a particular format for each of the received identified native audits;

compare each of the received identified native audits against at least one template and determine if each of the native audits matches at least one template; and

reducing each of the matched audits.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 826

FIG. 1

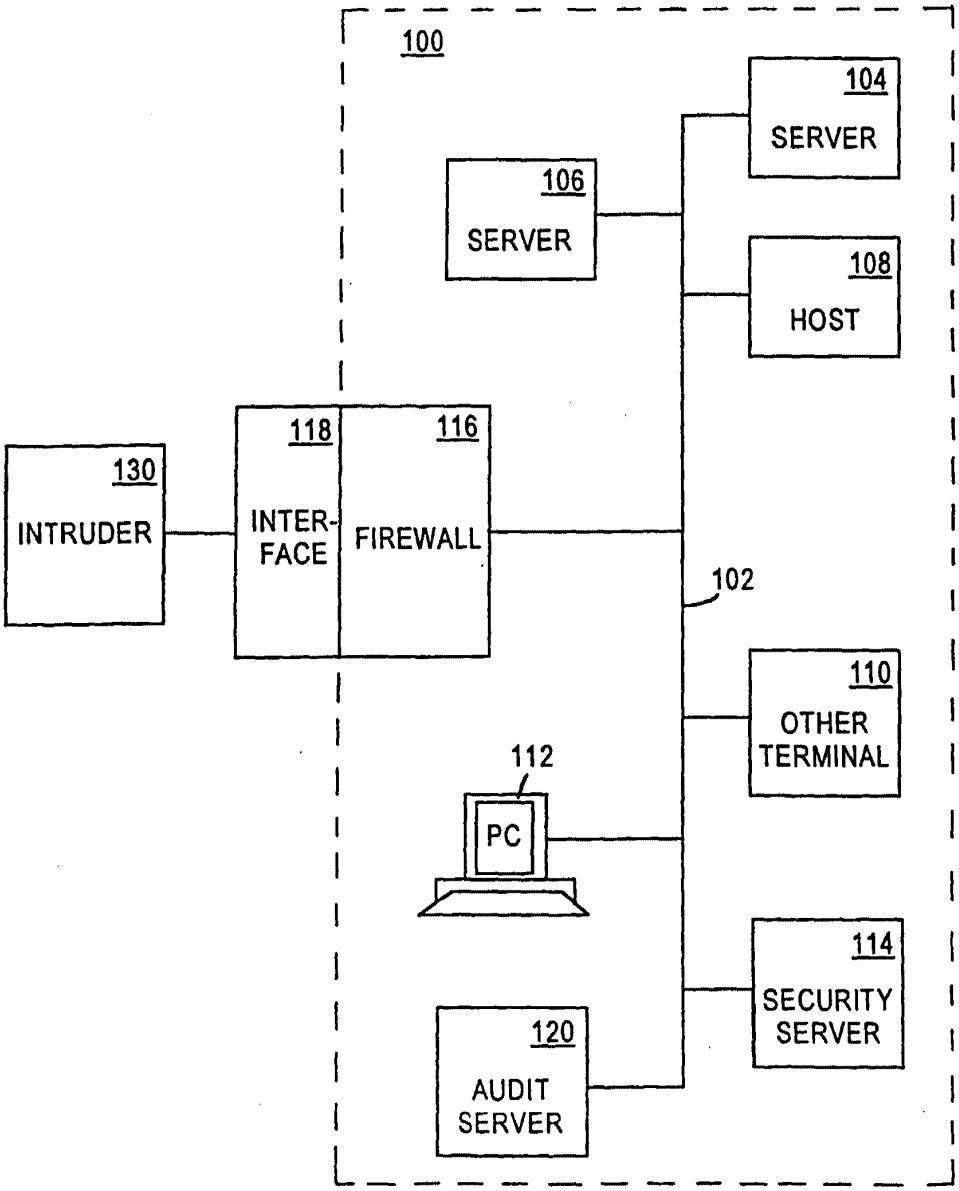

100
104
SERVER
106
SERVER
108
HOST
118
116
130
INTRUDER
INTER-
FACE
FIREWALL
102
110
OTHER
TERMINAL
112
PC
114
SECURITY
SERVER
120
AUDIT
SERVER

FIG. 2

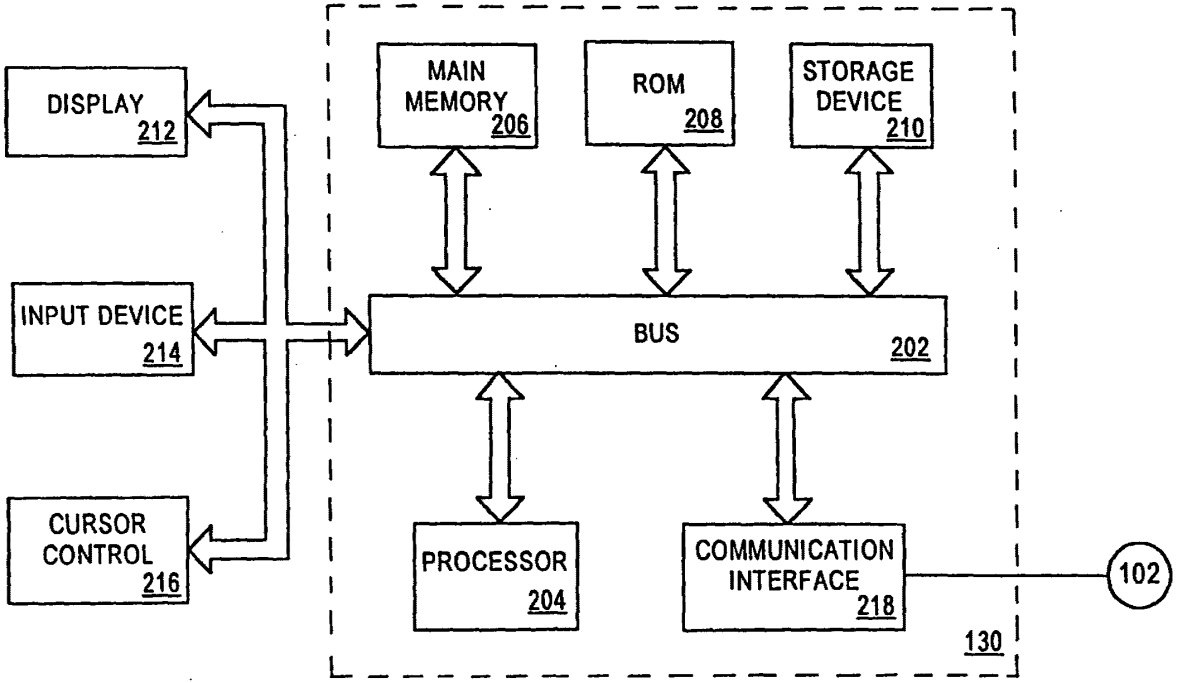
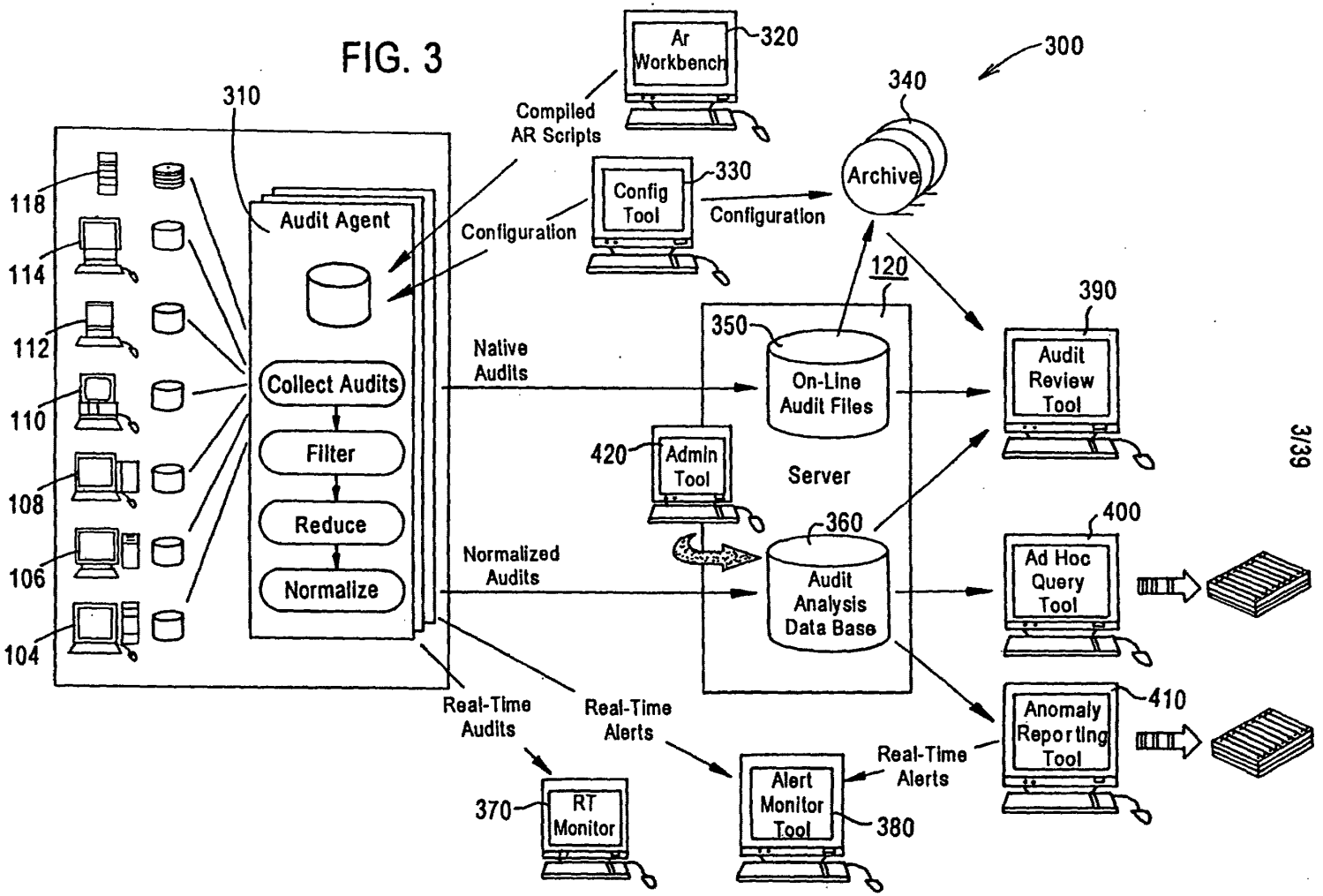

DISPLAY
212
INPUT DEVICE
214
CURSOR
CONTROL
216
MAIN
MEMORY
206
ROM
208
STORAGE
DEVICE
210
BUS
202
PROCESSOR
204
COMMUNICATION
INTERFACE
218
102
130

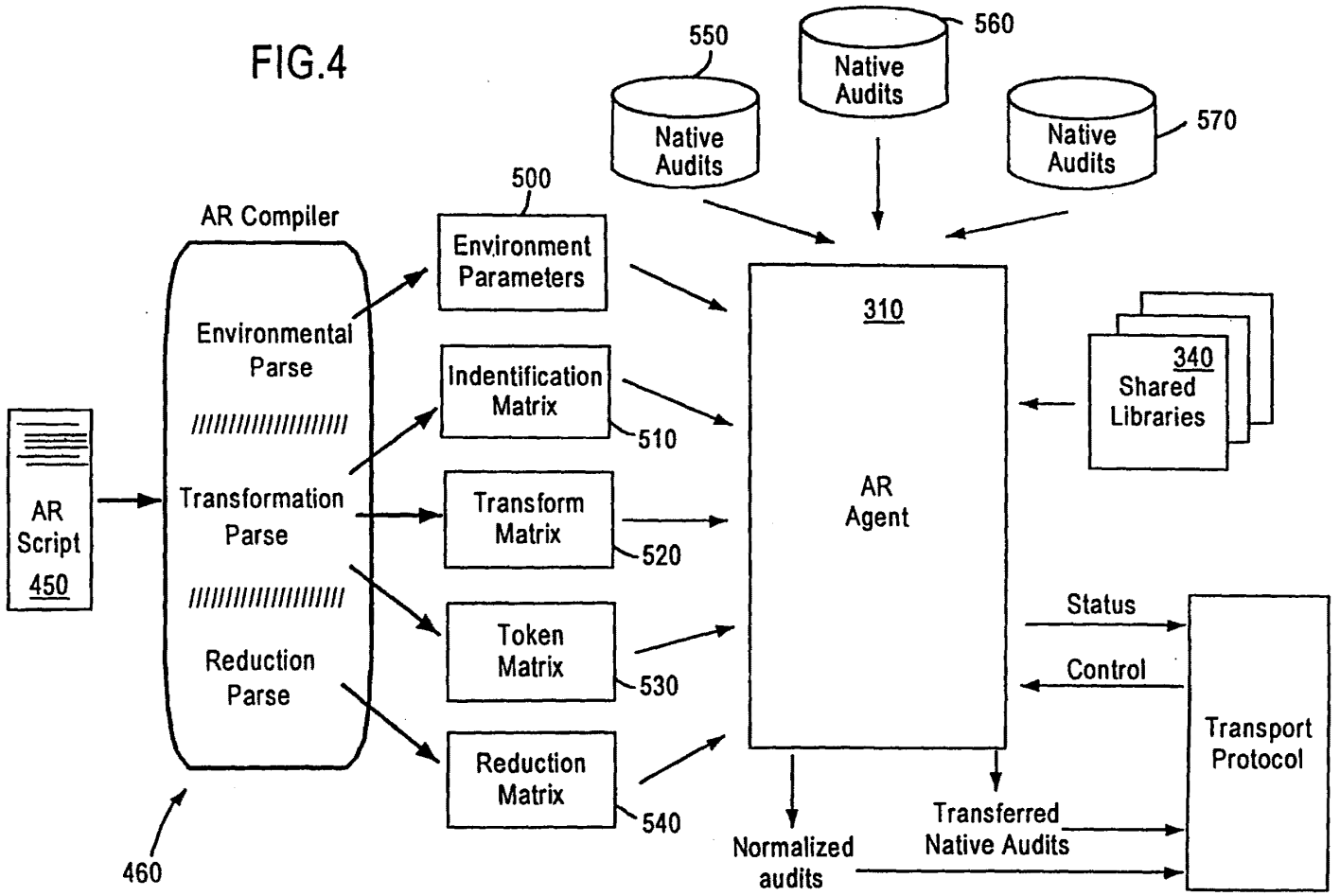
FIG. 3



WO 00/02115
PCT/US99/12181

IPR2024-00027
CrowdStrike Exhibit 1002 Page 829

FIG.4


550
560
Native Audits
Native Audits
Native Audits
570
500
AR Compiler
Environment Parameters
Environmental Parse
Indentification Matrix
510
Transformation Parse
Transform Matrix
520
Token Matrix
530
Reduction Parse
Reduction Matrix
540
AR Script
450
460
310
AR Agent
340
Shared Libraries
Status
Control
Transport Protocol
Normalized audits
Transferred Native Audits

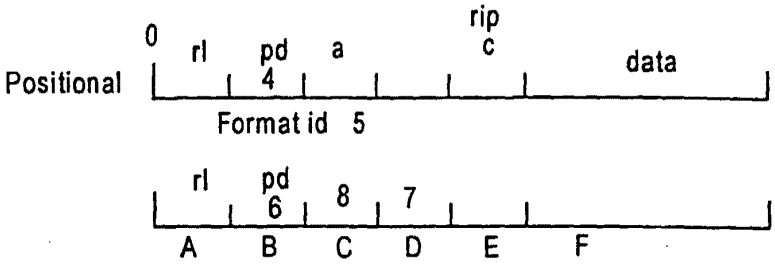

0
rl
pd
4
a
rip
c
data
Positional
Format id 5
rl
pd
6
8
7
A
B
C
D
E
F

FIG. 5A

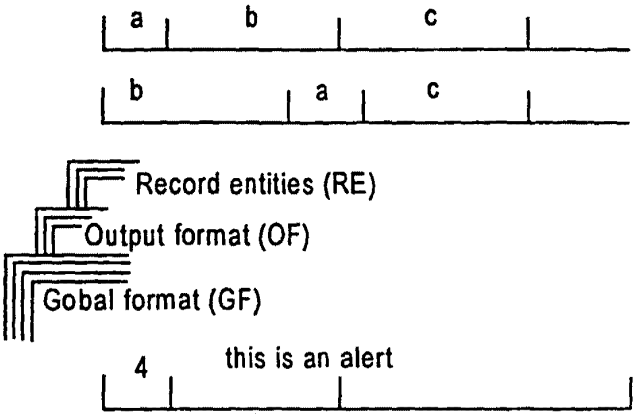

a
b
c
b
a
c
Record entities (RE)
Output format (OF)
Gobal format (GF)
4
this is an alert

FIG. 5B

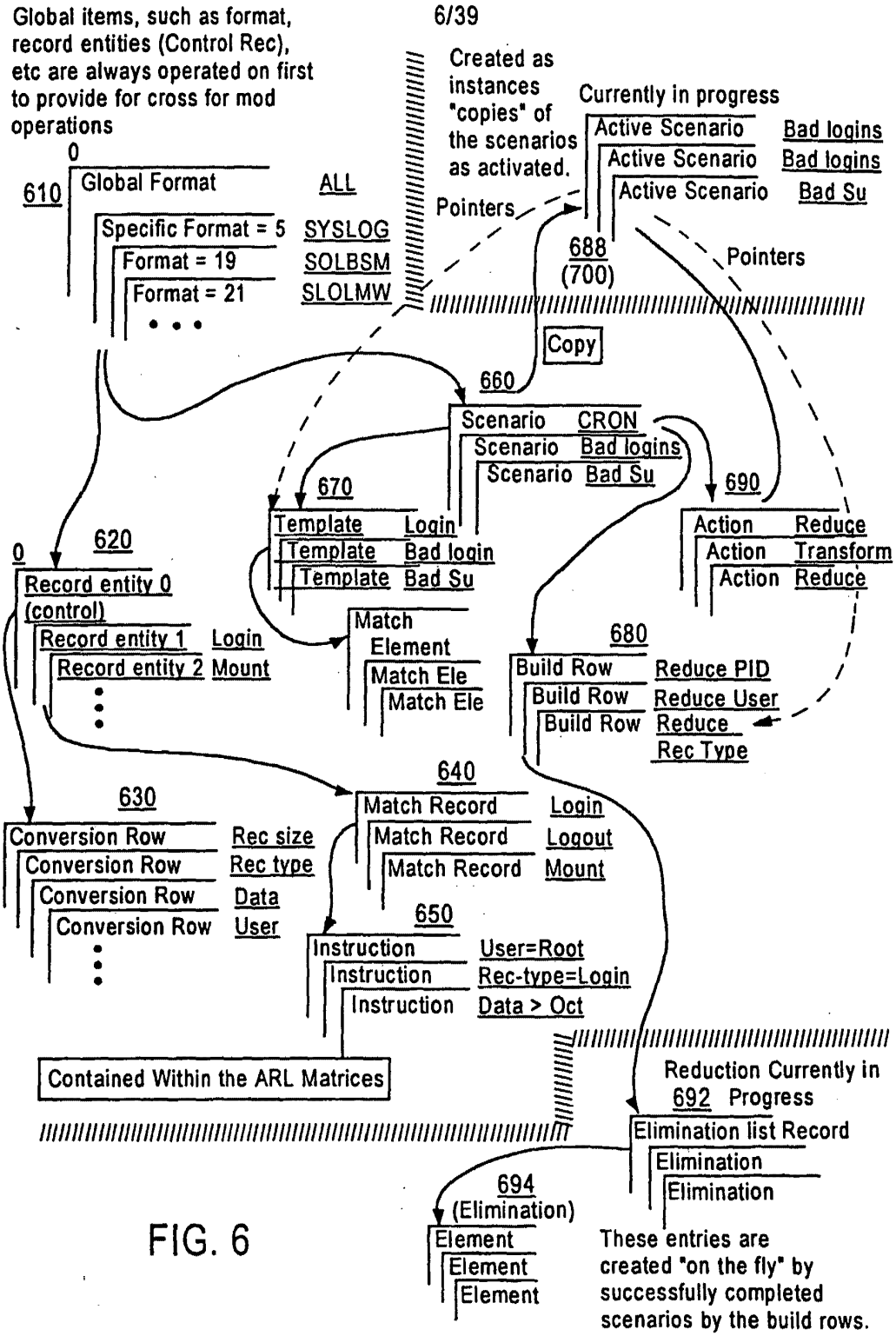

Global items, such as format, record entities (Control Rec), etc are always operated on first to provide for cross for mod operations
0
610
Global Format ALL
Specific Format = 5 SYSLOG
Format = 19 SOLBSM
Format = 21 SLOLMW
Created as instances "copies" of the scenarios as activated.
Pointers
Currently in progress
Active Scenario Bad logins
Active Scenario Bad logins
Active Scenario Bad Su
688
(700)
Pointers
Copy
660
Scenario CRON
Scenario Bad logins
Scenario Bad Su
670
Template Login
Template Bad login
Template Bad Su
690
Action Reduce
Action Transform
Action Reduce
0
620
Record entity 0 (control)
Record entity 1 Login
Record entity 2 Mount
Match Element
Match Ele
Match Ele
680
Build Row Reduce PID
Build Row Reduce User
Build Row Reduce Rec Type
640
630
Conversion Row Rec size
Conversion Row Rec type
Conversion Row Data
Conversion Row User
Match Record Login
Match Record Logout
Match Record Mount
650
Instruction User=Root
Instruction Rec-type=Login
Instruction Data > Oct
Contained Within the ARL Matrices
Reduction Currently in
692 Progress
Elimination list Record
Elimination
Elimination
694
(Elimination)
Element
Element
Element
These entries are created "on the fly" by successfully completed scenarios by the build rows.

FIG. 6

FIG. 7A

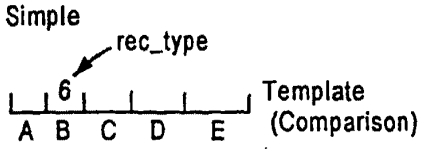


FIG. 7B

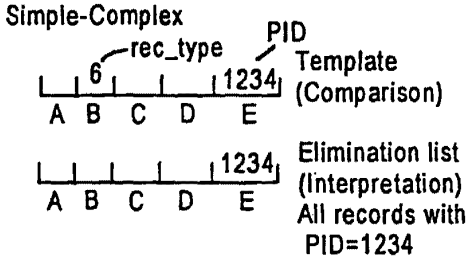


FIG. 7C

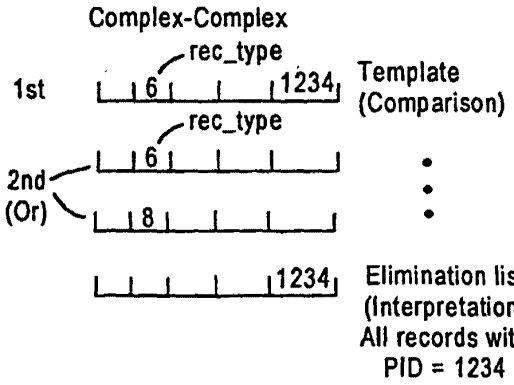
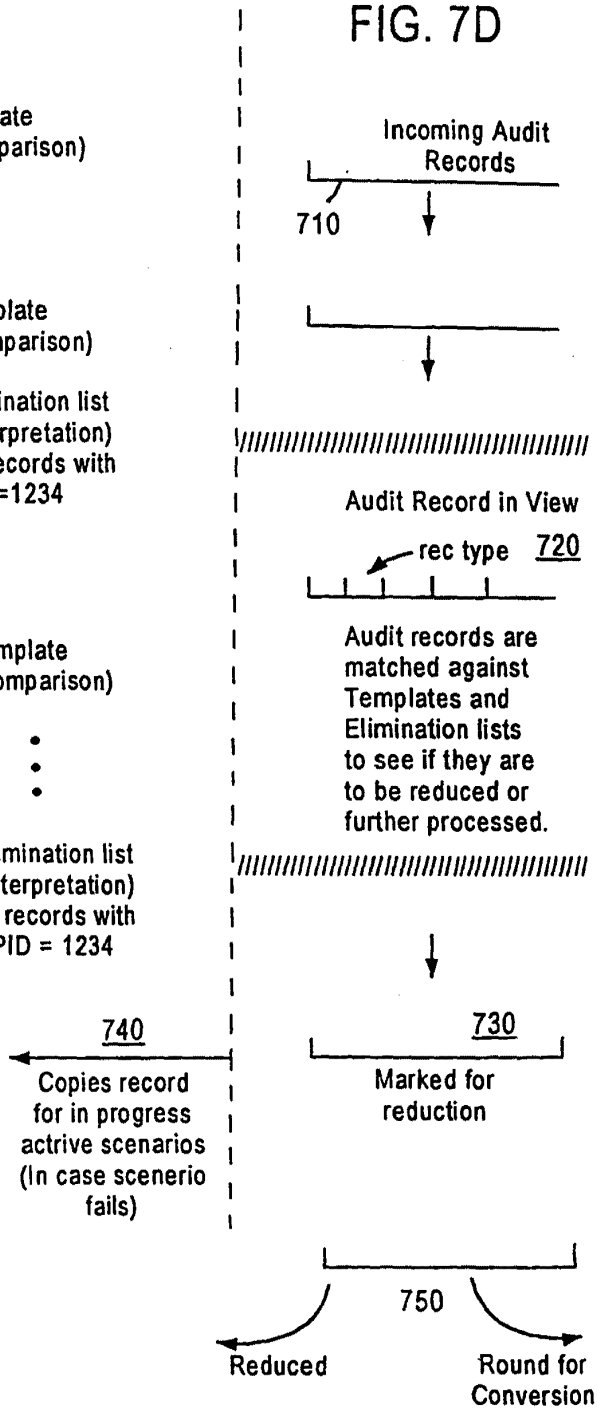


FIG. 7D



IPR2024-00027
CrowdStrike Exhibit 1002 Page 833

Fig. 8

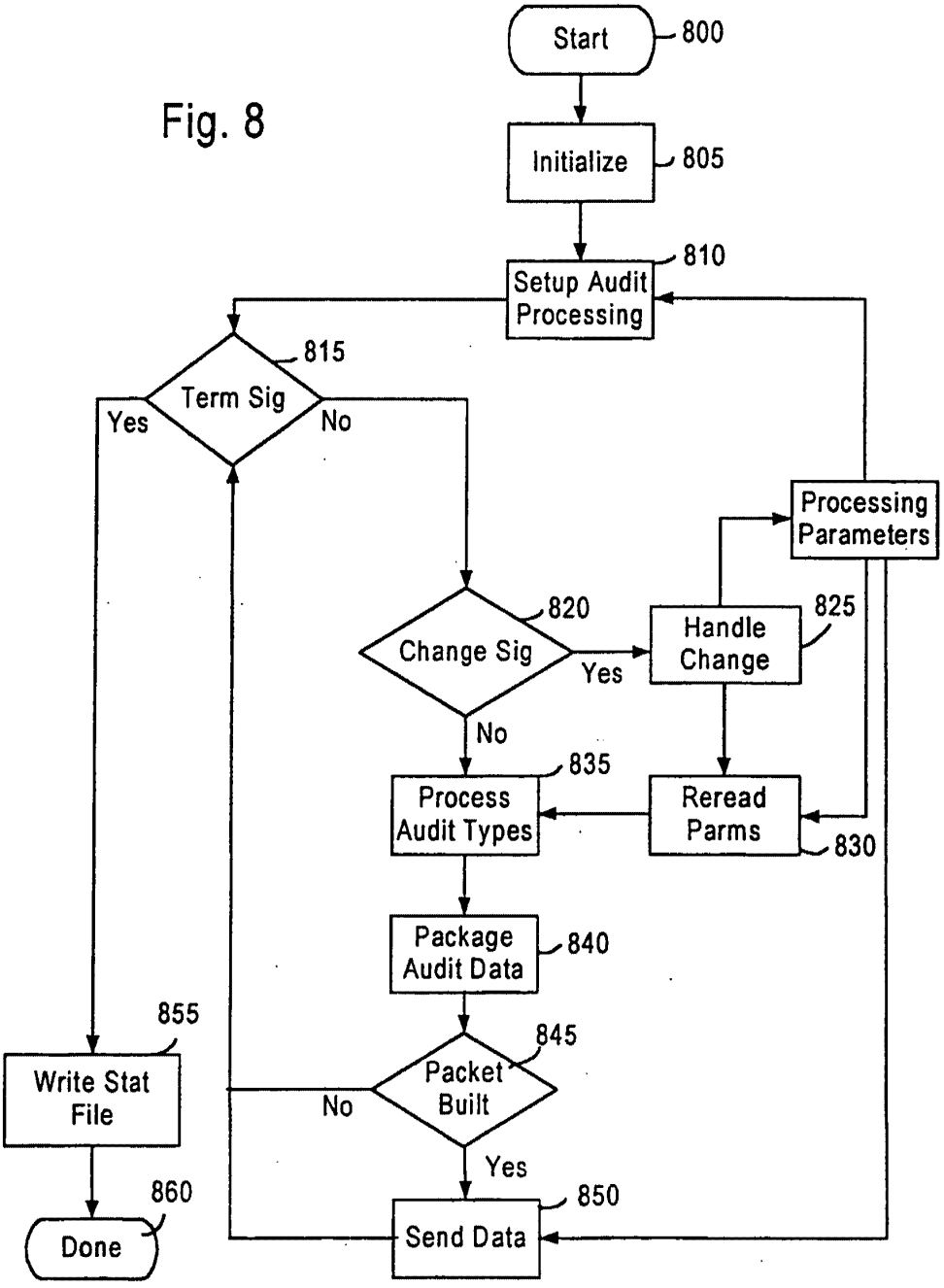
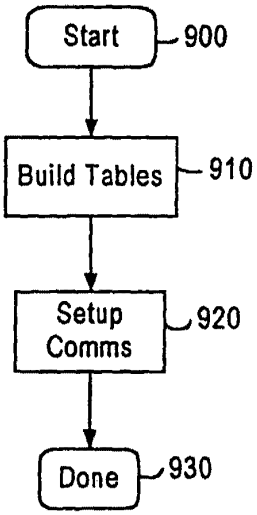

Start
800
Initialize
805
Setup Audit Processing
810
Term Sig
815
Yes
No
Processing Parameters
Change Sig
820
Yes
Handle Change
825
No
Process Audit Types
835
Reread Parms
830
Package Audit Data
840
Write Stat File
855
Packet Built
845
No
Yes
Done
860
Send Data
850

FIG. 9



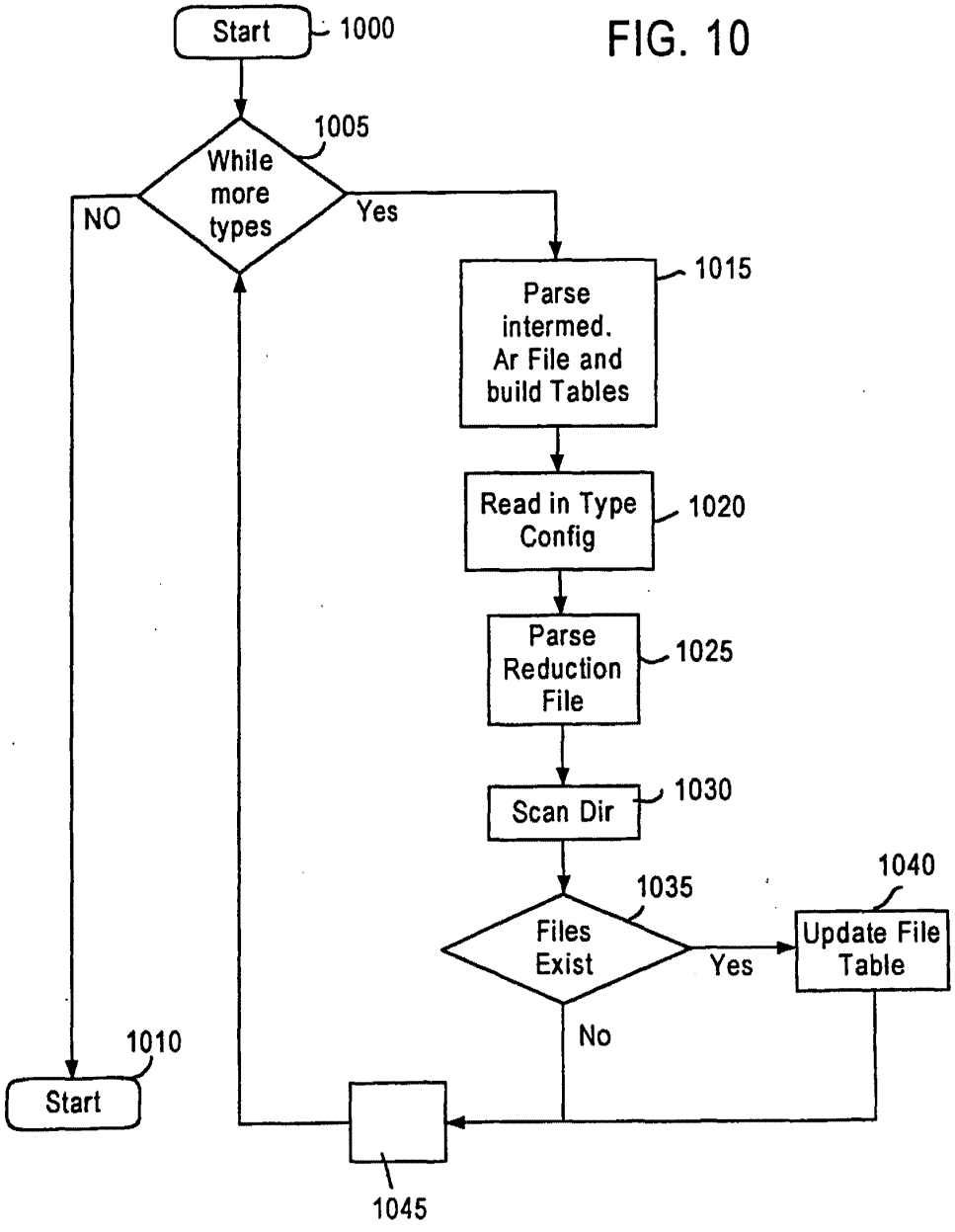
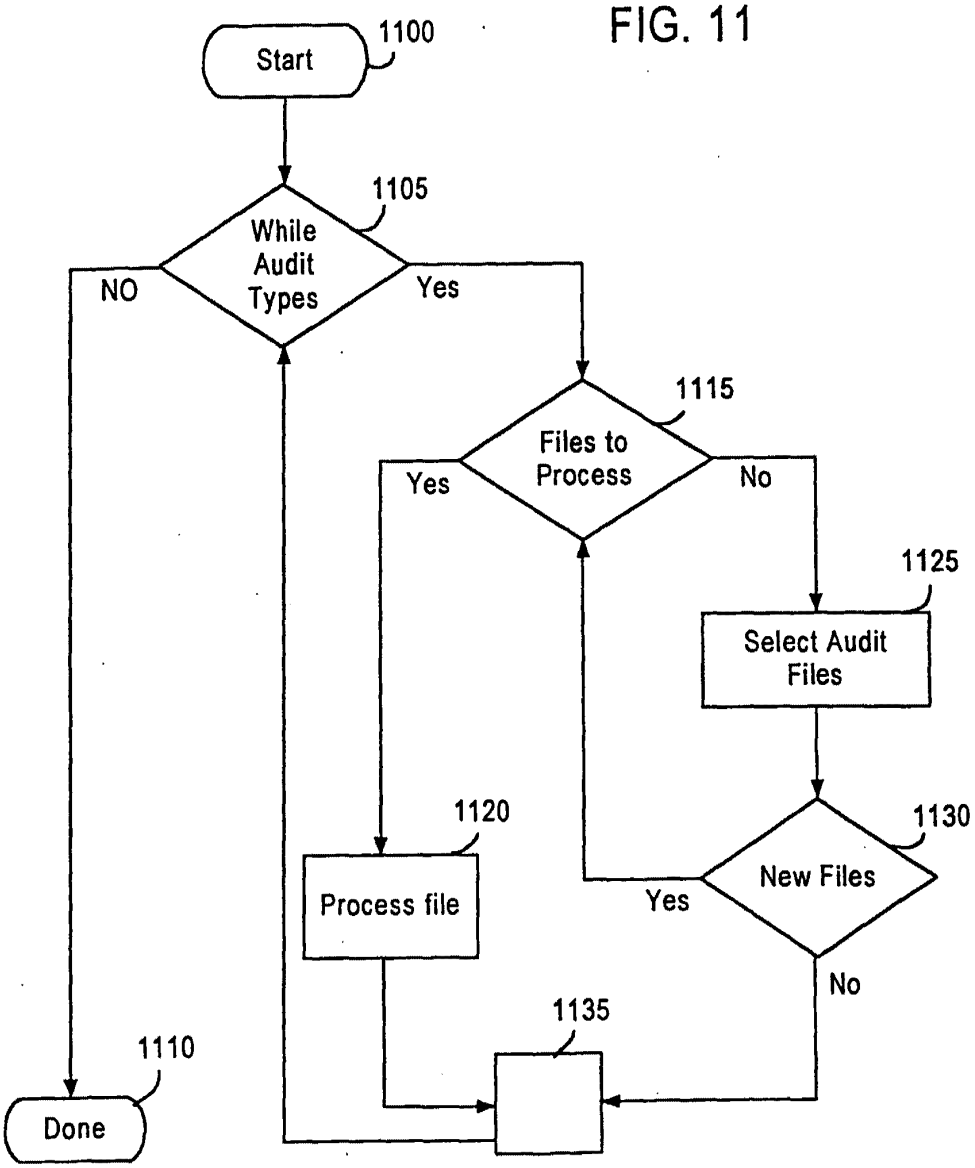

Start
1000
While more types
1005
NO
Yes
Parse intermed. Ar File and build Tables
1015
Read in Type Config
1020
Parse Reduction File
1025
Scan Dir
1030
Files Exist
1035
Update File Table
1040
No
Start
1010
1045

FIG. 10

FIG. 11


Start
1100
While Audit Types
1105
NO
Yes
Files to Process
1115
Yes
No
Select Audit Files
1125
Process file
1120
New Files
1130
Yes
No
1135
Done
1110

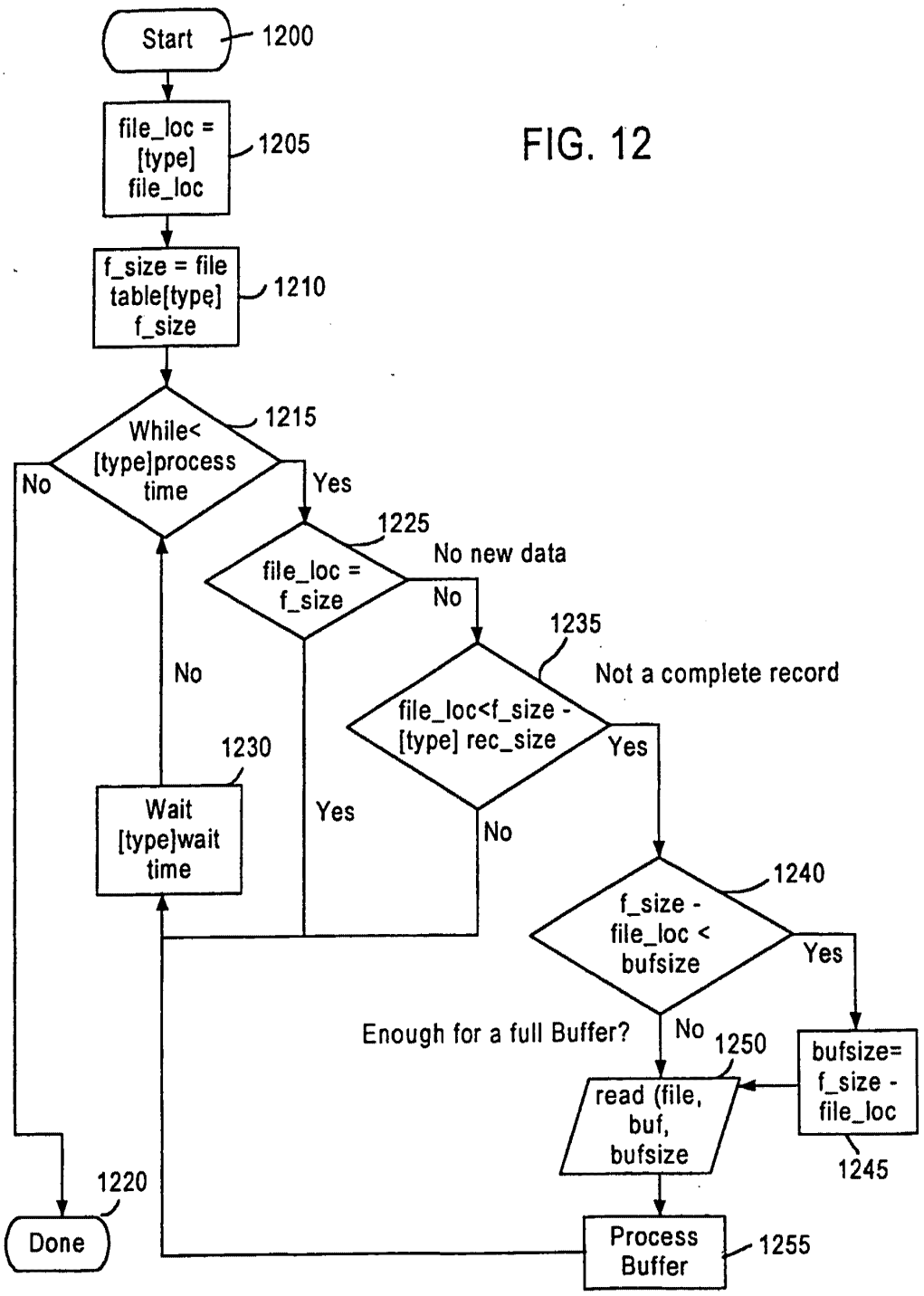

Start
1200
file_loc = [type] file_loc
1205
f_size = file table[type] f_size
1210
While< [type]process time
1215
No
Yes
file_loc = f_size
1225
No new data
No
1235
Not a complete record
file_loc<f_size - [type] rec_size
Yes
No
Yes
No
1230
Wait [type]wait time
f_size - file_loc < bufsize
1240
Yes
Enough for a full Buffer?
No
1250
bufsize= f_size - file_loc
1245
read (file, buf, bufsize
Process Buffer
1255
1220
Done

FIG. 12

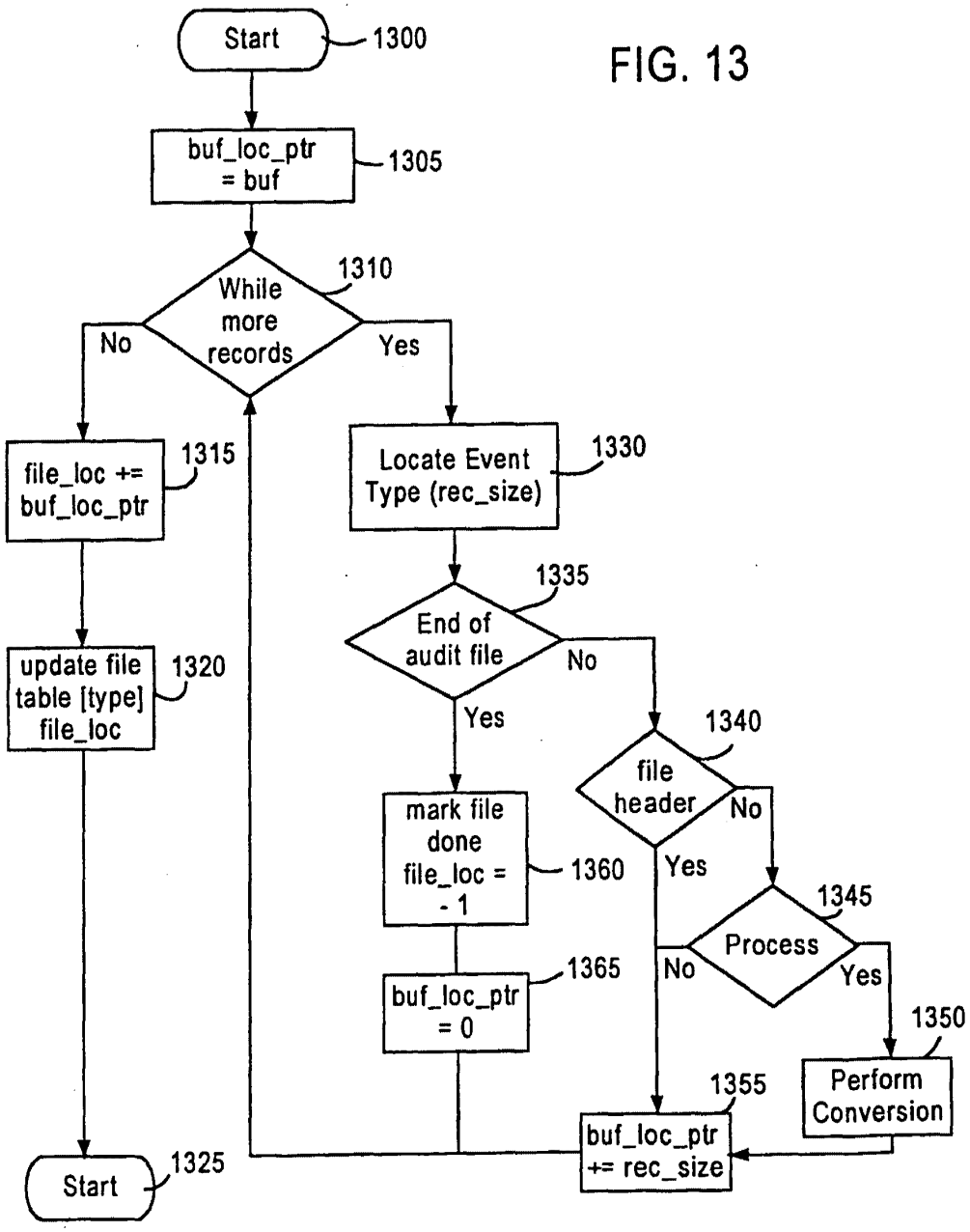
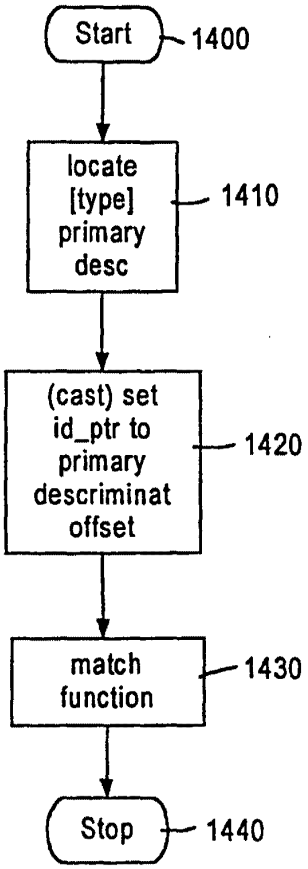

Start
1300
buf_loc_ptr = buf
1305
While more records
1310
No
Yes
file_loc += buf_loc_ptr
1315
update file table [type] file_loc
1320
Start
1325
Locate Event Type (rec_size)
1330
End of audit file
1335
No
Yes
mark file done file_loc = - 1
1360
buf_loc_ptr = 0
1365
file header
1340
No
Yes
Process
1345
No
Yes
Perform Conversion
1350
buf_loc_ptr += rec_size
1355

FIG. 13

FIG. 14


Start
1400
locate [type] primary desc
1410
(cast) set id_ptr to primary descriminat offset
1420
match function
1430
Stop
1440

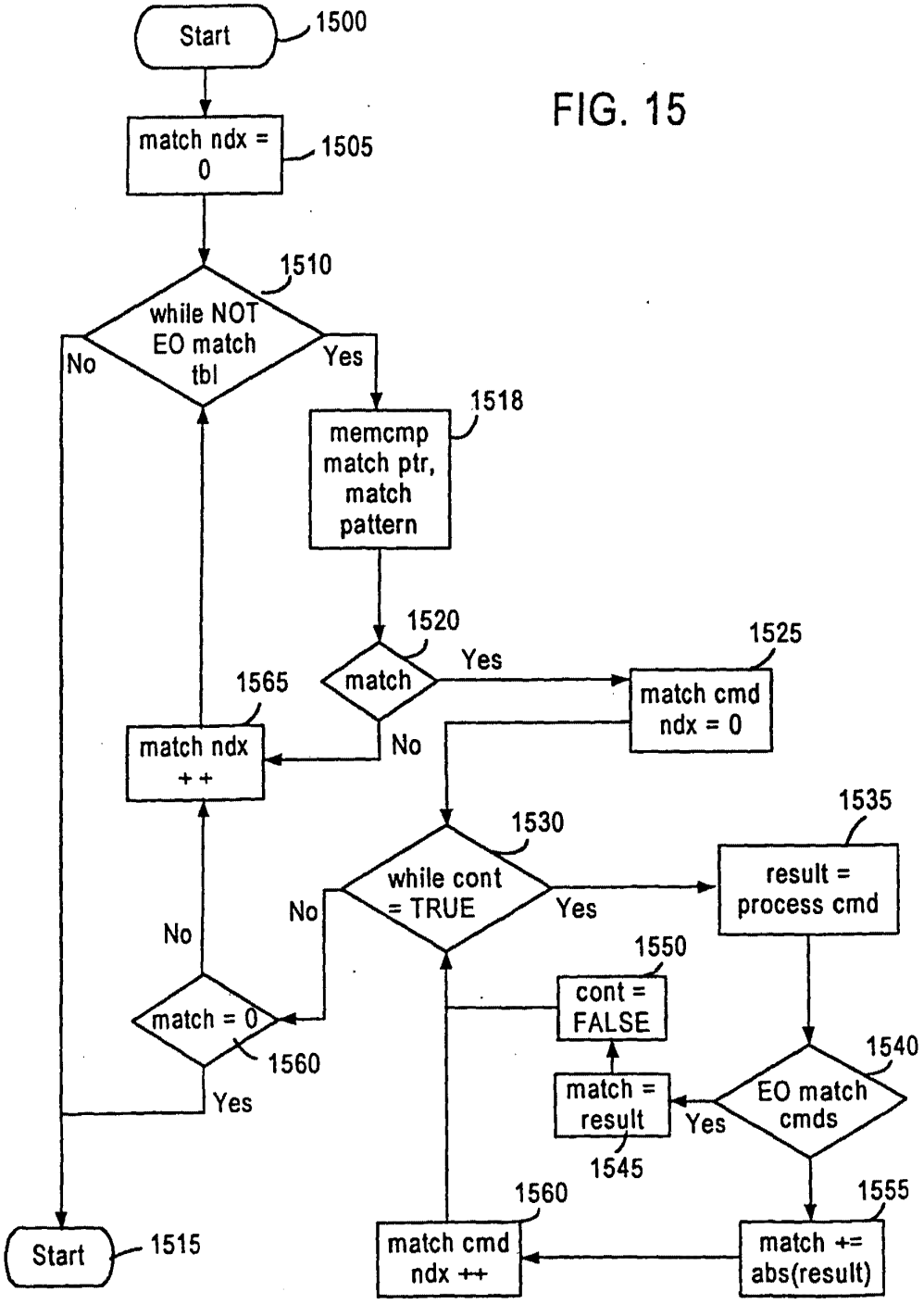

Start
1500
match ndx = 0
1505
1510
while NOT EO match tbl
No
Yes
1518
memcmp match ptr, match pattern
1520
match
Yes
1525
match cmd ndx = 0
1565
match ndx ++
No
1530
while cont = TRUE
No
Yes
1535
result = process cmd
No
1550
cont = FALSE
match = 0
1560
Yes
1540
EO match cmds
match = result
Yes
1545
1555
match += abs(result)
1560
match cmd ndx ++
Start
1515

FIG. 15

FIG. 16

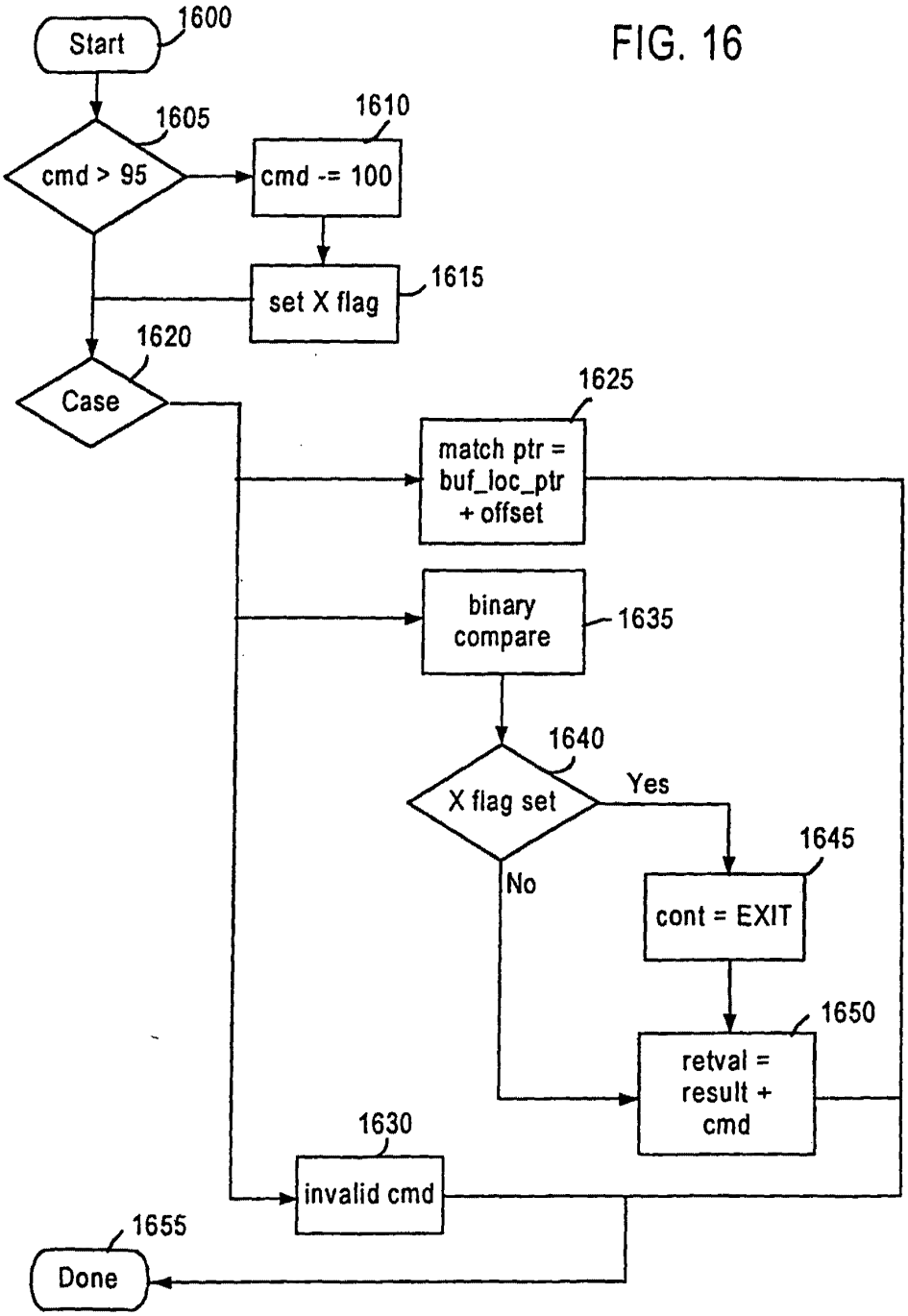

Start
1600
1605
cmd > 95
1610
cmd -= 100
set X flag
1615
1620
Case
1625
match ptr = buf_loc_ptr + offset
binary compare
1635
1640
X flag set
Yes
No
1645
cont = EXIT
1650
retval = result + cmd
1630
invalid cmd
1655
Done

FIG.17

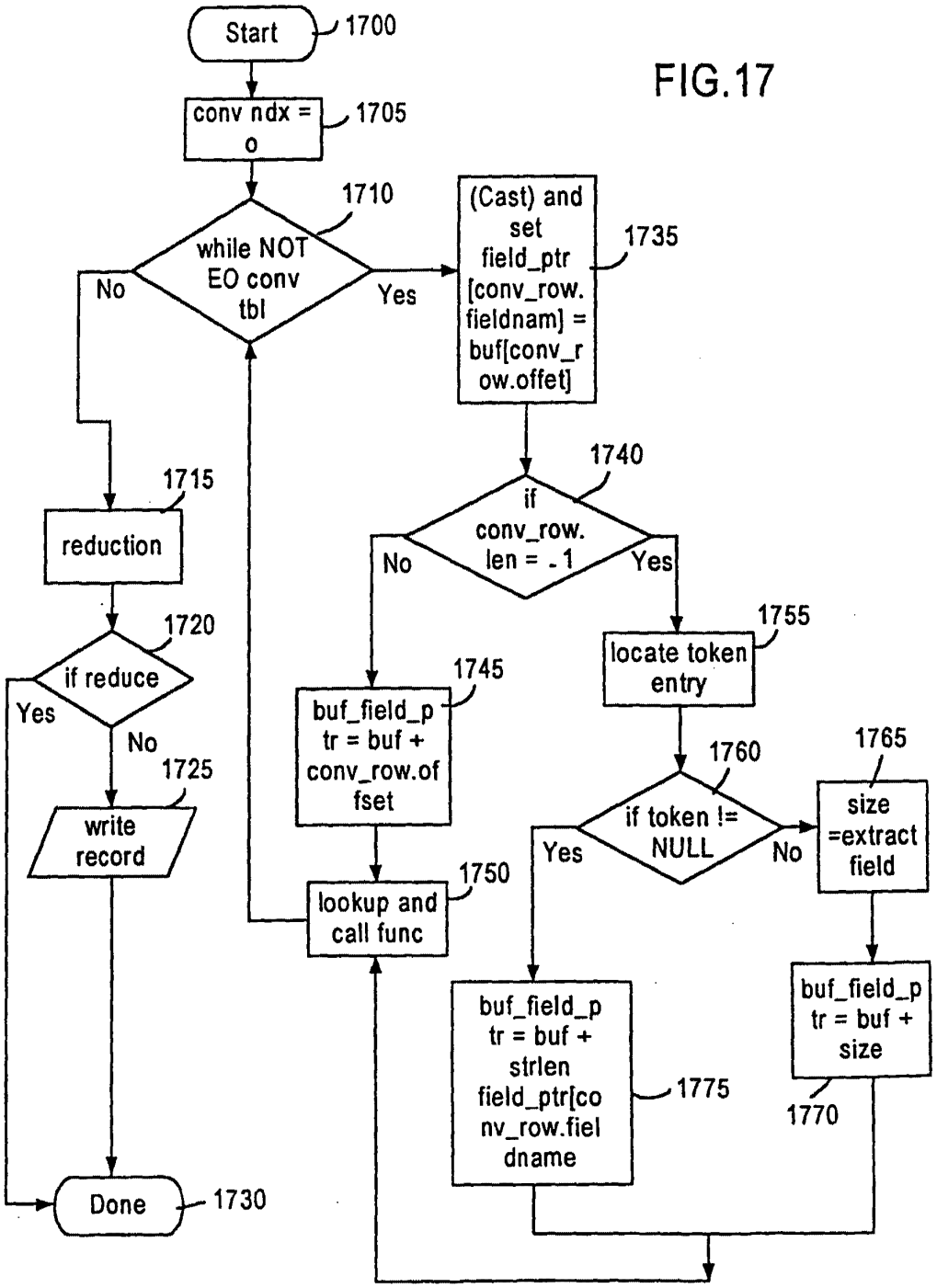

Start
1700
conv ndx = o
1705
while NOT EO conv tbl
1710
No
Yes
(Cast) and set field_ptr [conv_row. fieldnam] = buf[conv_r ow.offet]
1735
reduction
1715
if conv_row. len = -1
1740
No
Yes
if reduce
1720
Yes
No
write record
1725
buf_field_p tr = buf + conv_row.of fset
1745
locate token entry
1755
if token != NULL
1760
Yes
No
size =extract field
1765
lookup and call func
1750
buf_field_p tr = buf + strlen field_ptr[co nv_row.fiel dname
1775
buf_field_p tr = buf + size
1770
Done
1730

FIG. 18

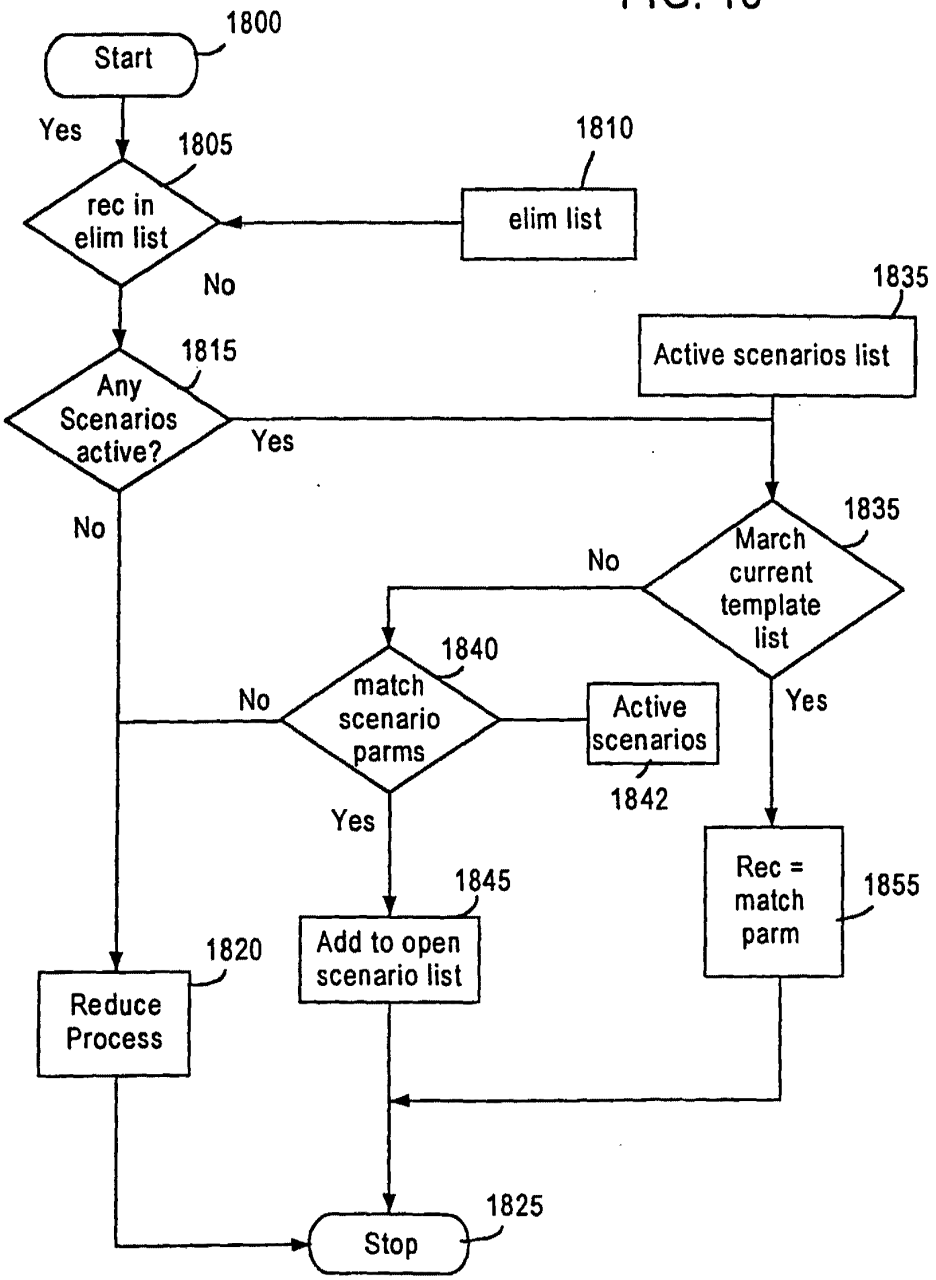
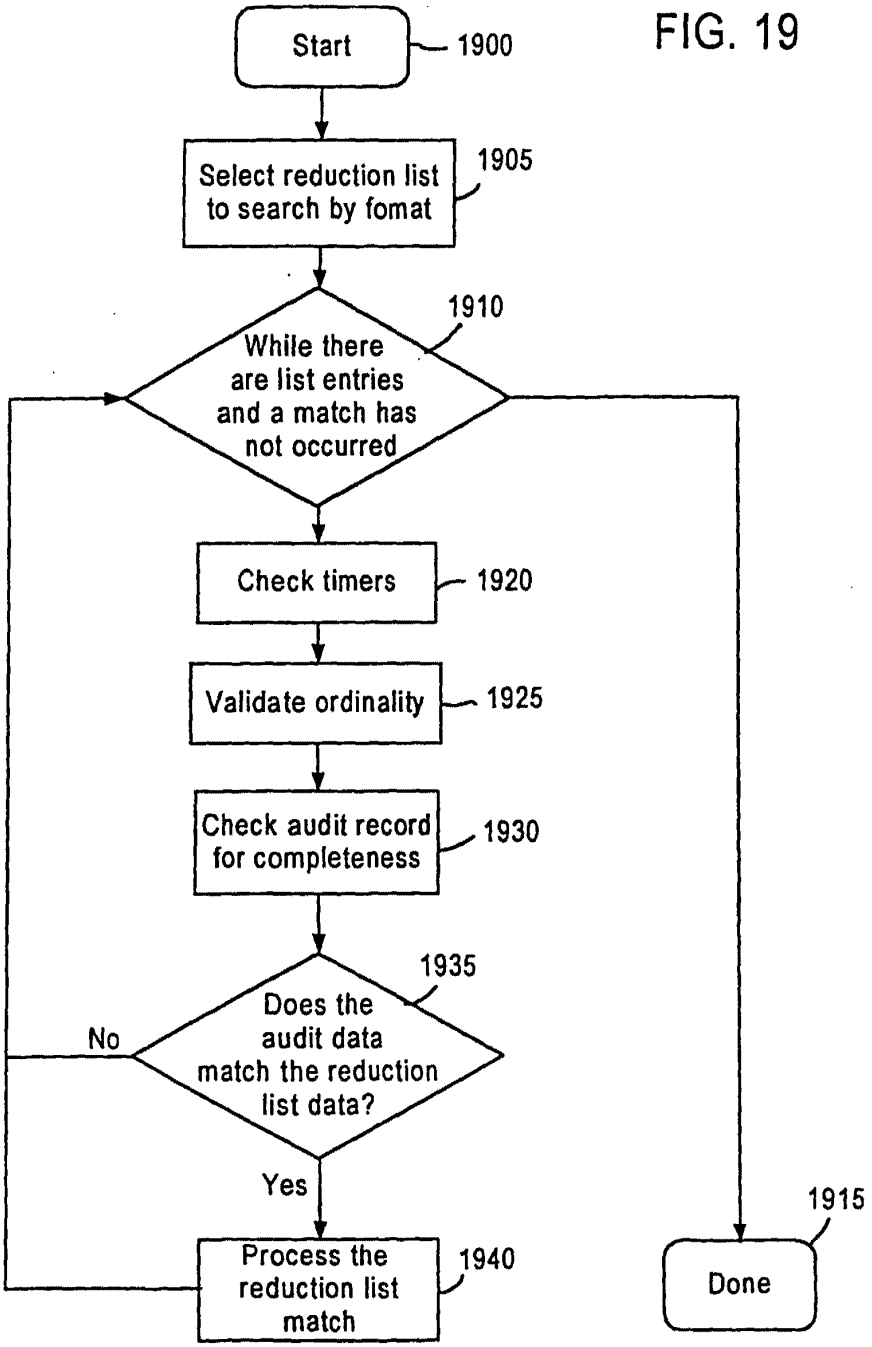

1800
Start
Yes
1805
rec in elim list
1810
elim list
No
1835
Active scenarios list
1815
Any Scenarios active?
Yes
No
1835
March current template list
No
Yes
1840
match scenario parms
No
Active scenarios
1842
Yes
1845
Add to open scenario list
Rec = match parm
1855
1820
Reduce Process
1825
Stop

FIG. 19



IPR2024-00027
CrowdStrike Exhibit 1002 Page 845

FIG.20

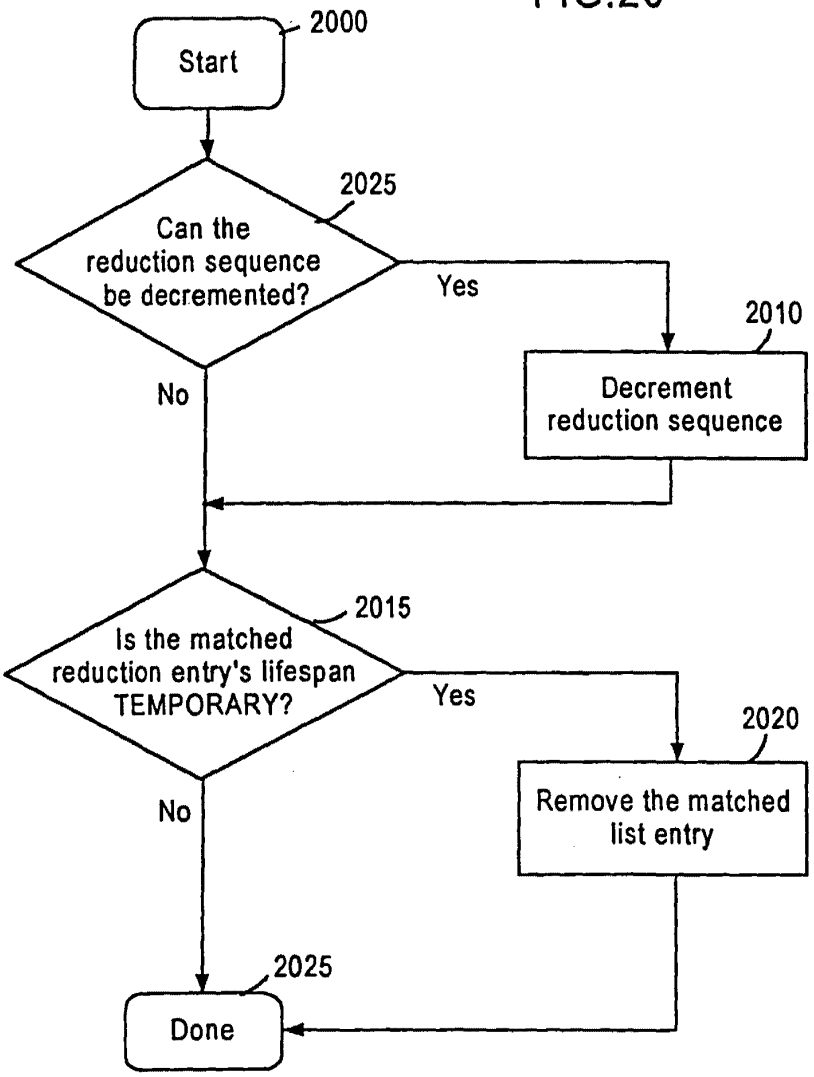

2000
Start
2025
Can the reduction sequence be decremented?
Yes
No
2010
Decrement reduction sequence
2015
Is the matched reduction entry's lifespan TEMPORARY?
Yes
No
2020
Remove the matched list entry
2025
Done

FIG. 21

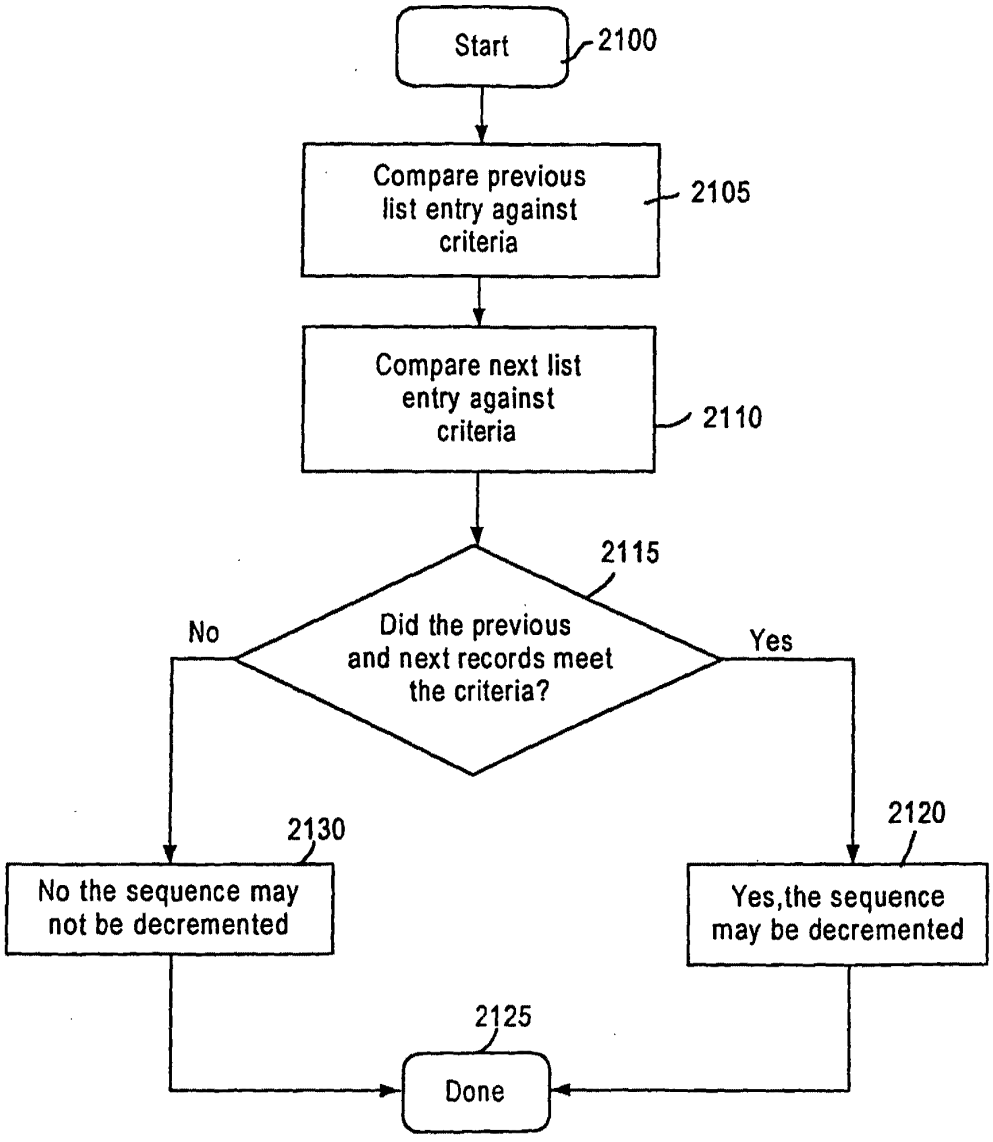

Start
2100
Compare previous list entry against criteria
2105
Compare next list entry against criteria
2110
2115
Did the previous and next records meet the criteria?
No
Yes
2130
No the sequence may not be decremented
2120
Yes,the sequence may be decremented
2125
Done

FIG. 22

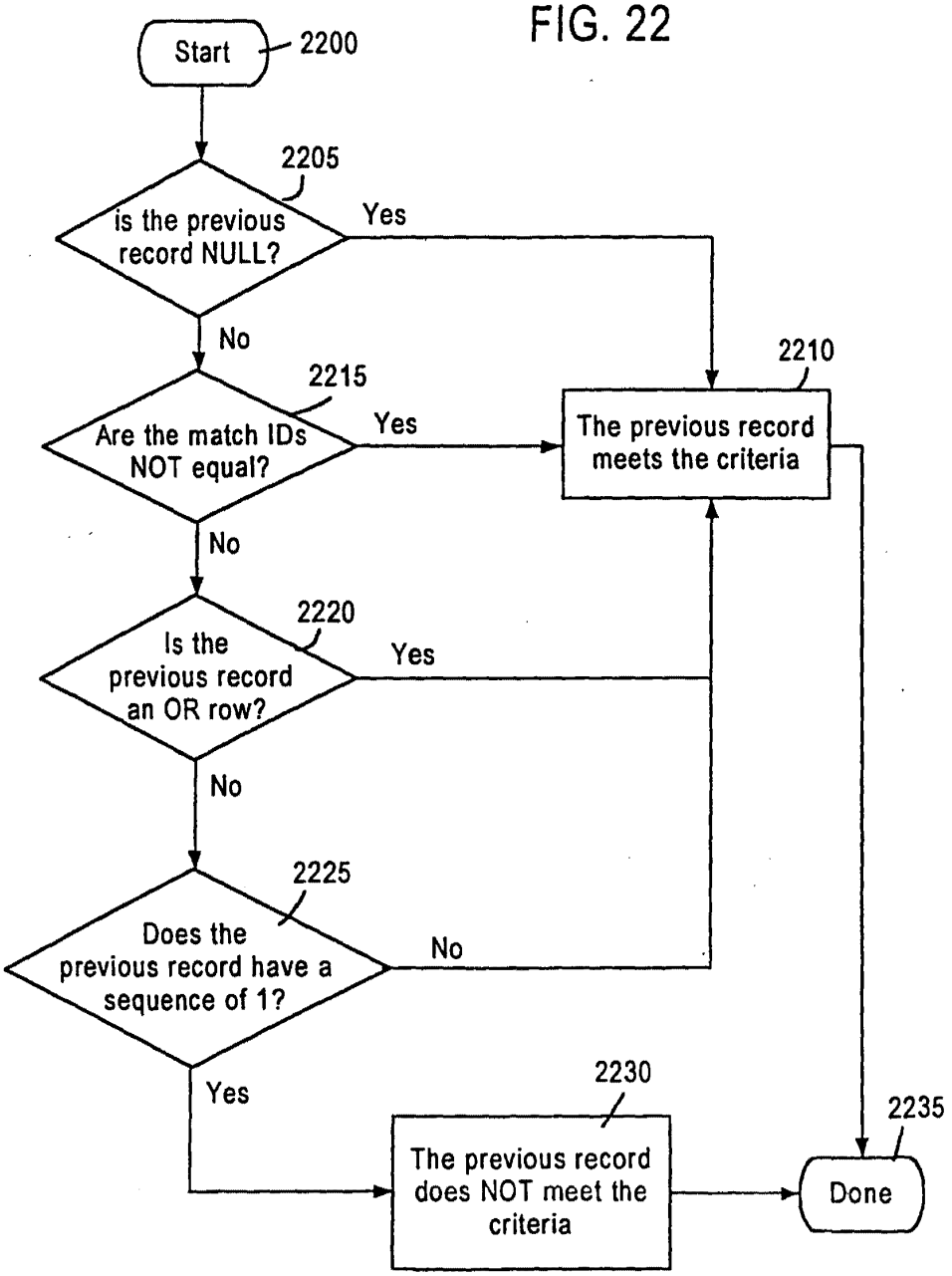

Start
2200
2205
is the previous record NULL?
Yes
No
2210
2215
Are the match IDs NOT equal?
Yes
The previous record meets the criteria
No
2220
Is the previous record an OR row?
Yes
No
2225
Does the previous record have a sequence of 1?
No
Yes
2230
The previous record does NOT meet the criteria
2235
Done

FIG. 23

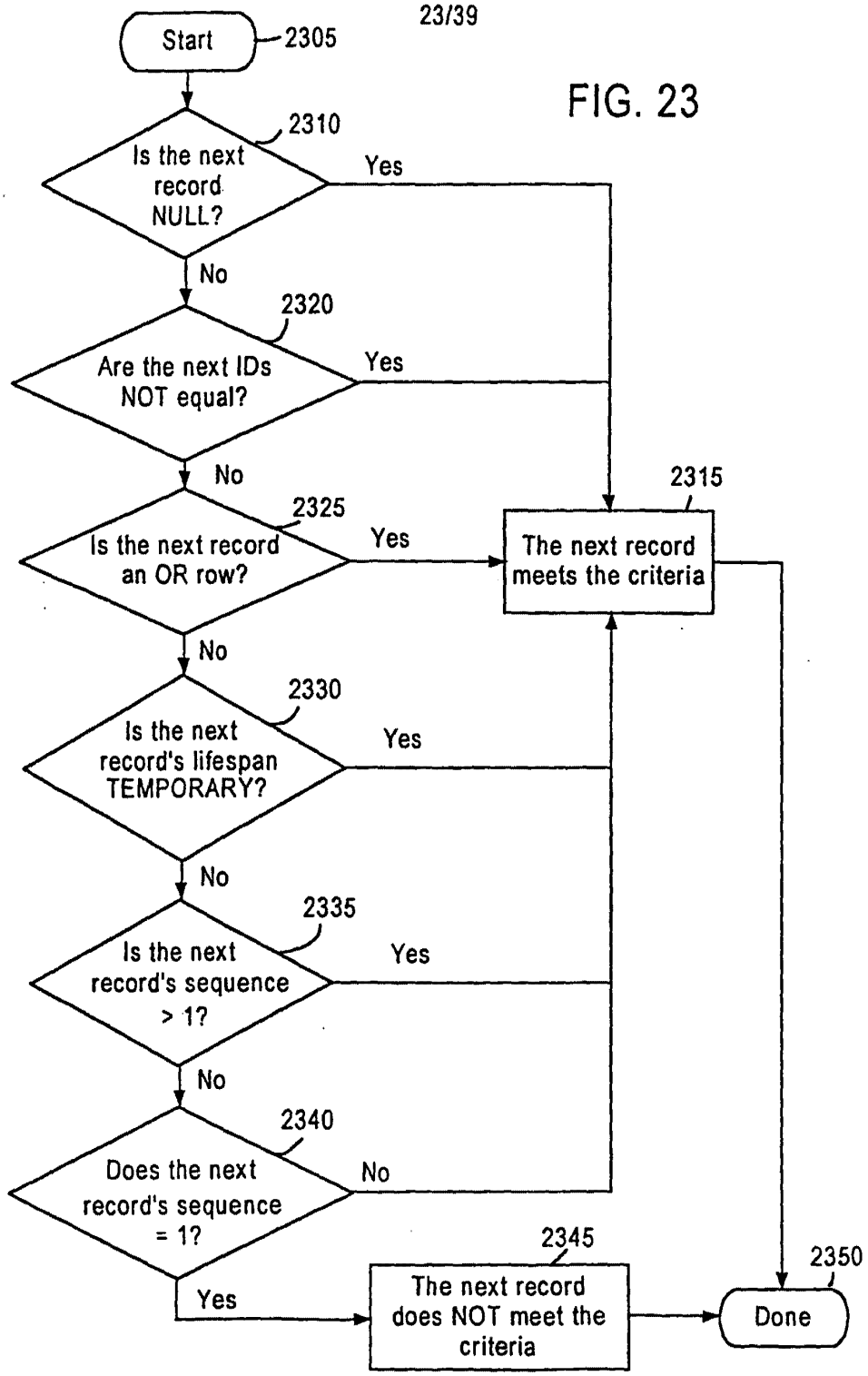

Start
2305
2310
Is the next record NULL?
Yes
No
2320
Are the next IDs NOT equal?
Yes
No
2325
Is the next record an OR row?
Yes
No
2330
Is the next record's lifespan TEMPORARY?
Yes
No
2335
Is the next record's sequence > 1?
Yes
No
2340
Does the next record's sequence = 1?
No
Yes
2315
The next record meets the criteria
2345
The next record does NOT meet the criteria
2350
Done

FIG. 24

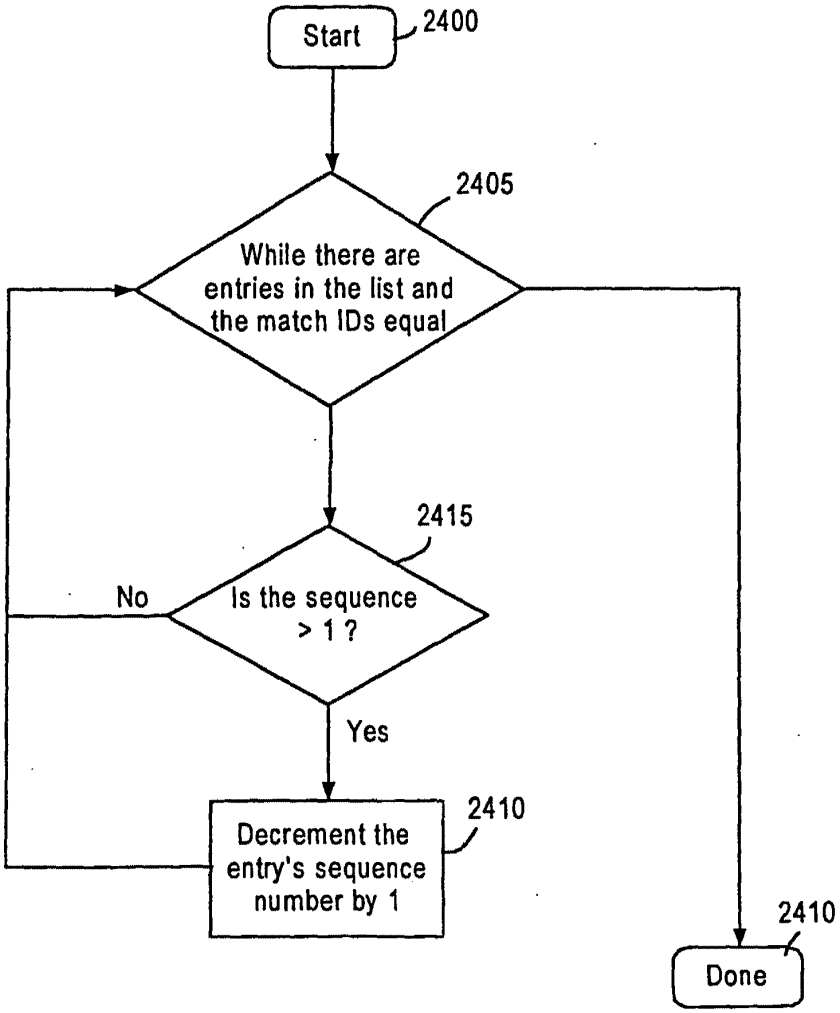

Start
2400
2405
While there are entries in the list and the match IDs equal
2415
No
Is the sequence > 1 ?
Yes
2410
Decrement the entry's sequence number by 1
2410
Done

FIG. 25

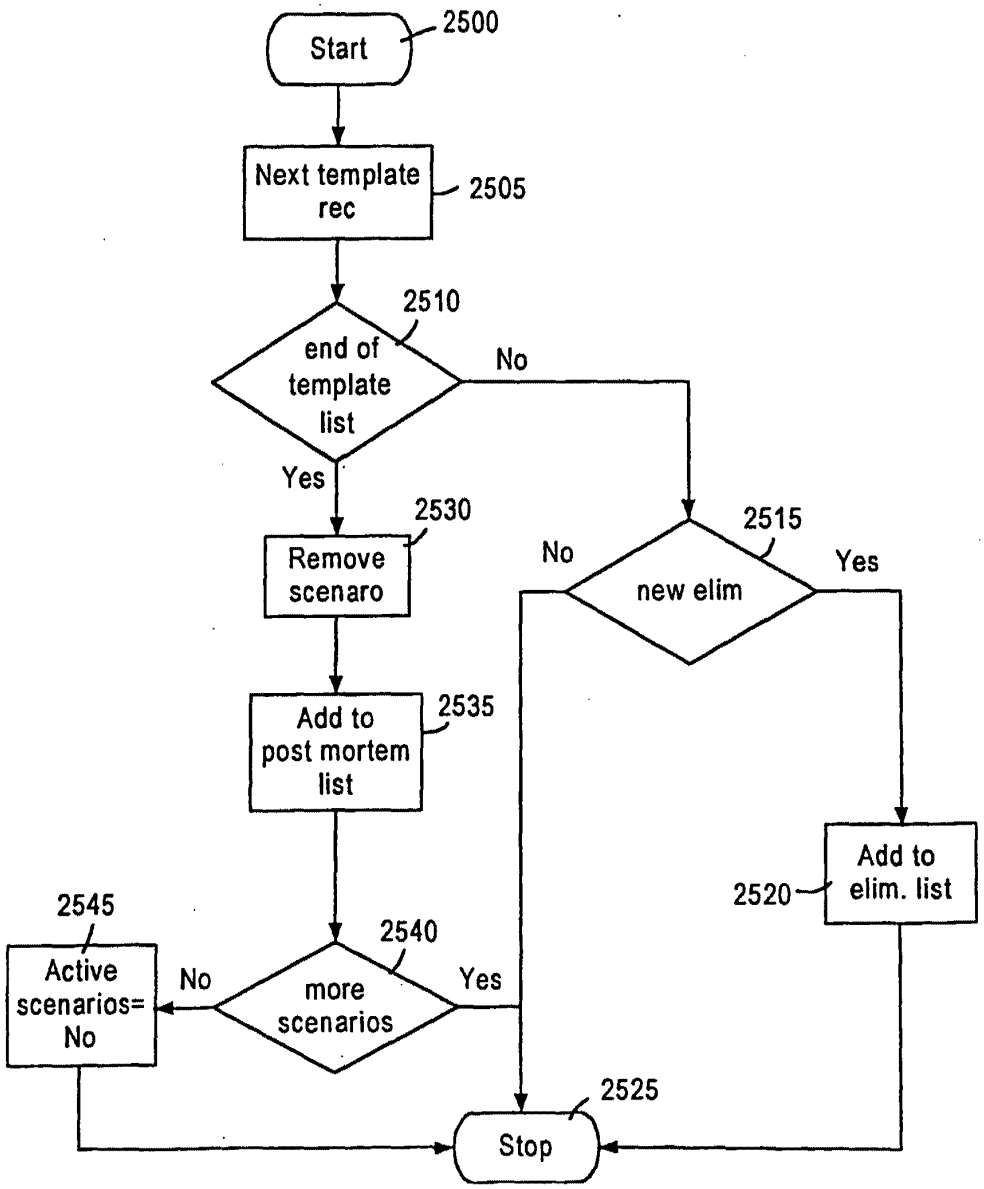

Start
2500
Next template rec
2505
end of template list
2510
No
Yes
Remove scenaro
2530
new elim
2515
No
Yes
Add to post mortem list
2535
Add to elim. list
2520
Active scenarios= No
2545
more scenarios
2540
No
Yes
Stop
2525

FIG. 26

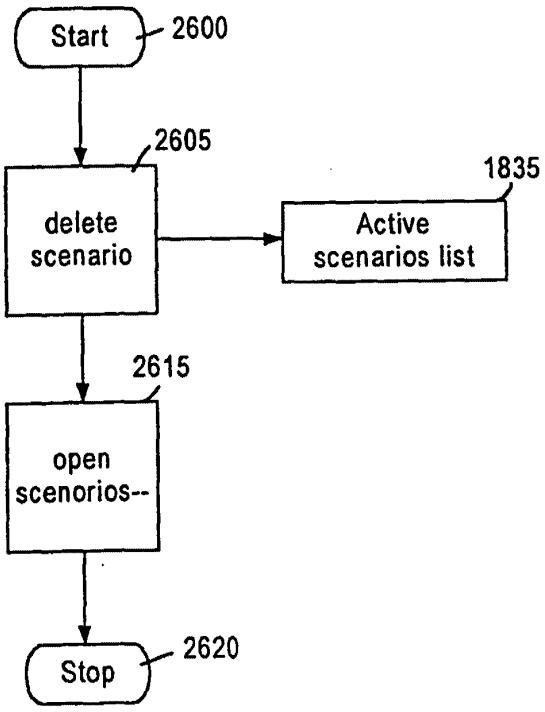

Start
2600
2605
delete scenario
1835
Active scenarios list
2615
open scenorios--
Stop
2620

FIG. 27

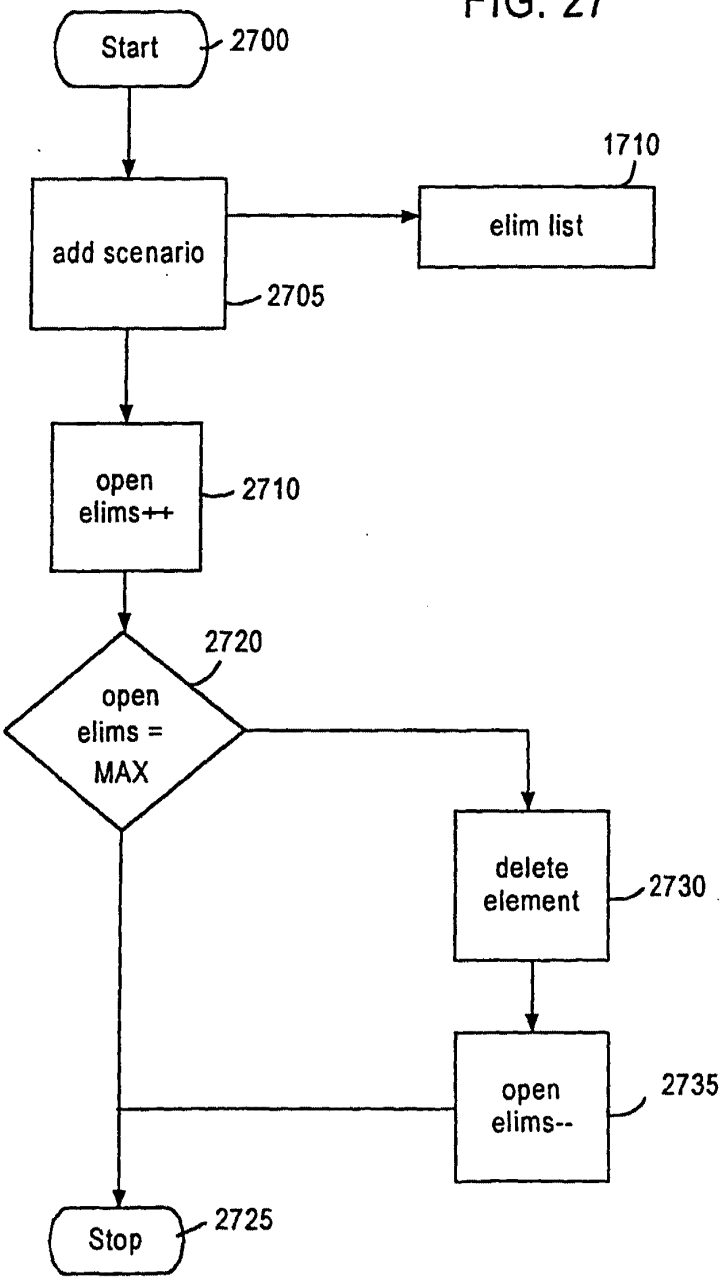

Start
2700
1710
add scenario
elim list
2705
open
elims++
2710
2720
open
elims =
MAX
delete
element
2730
open
elims--
2735
Stop
2725

FIG. 28

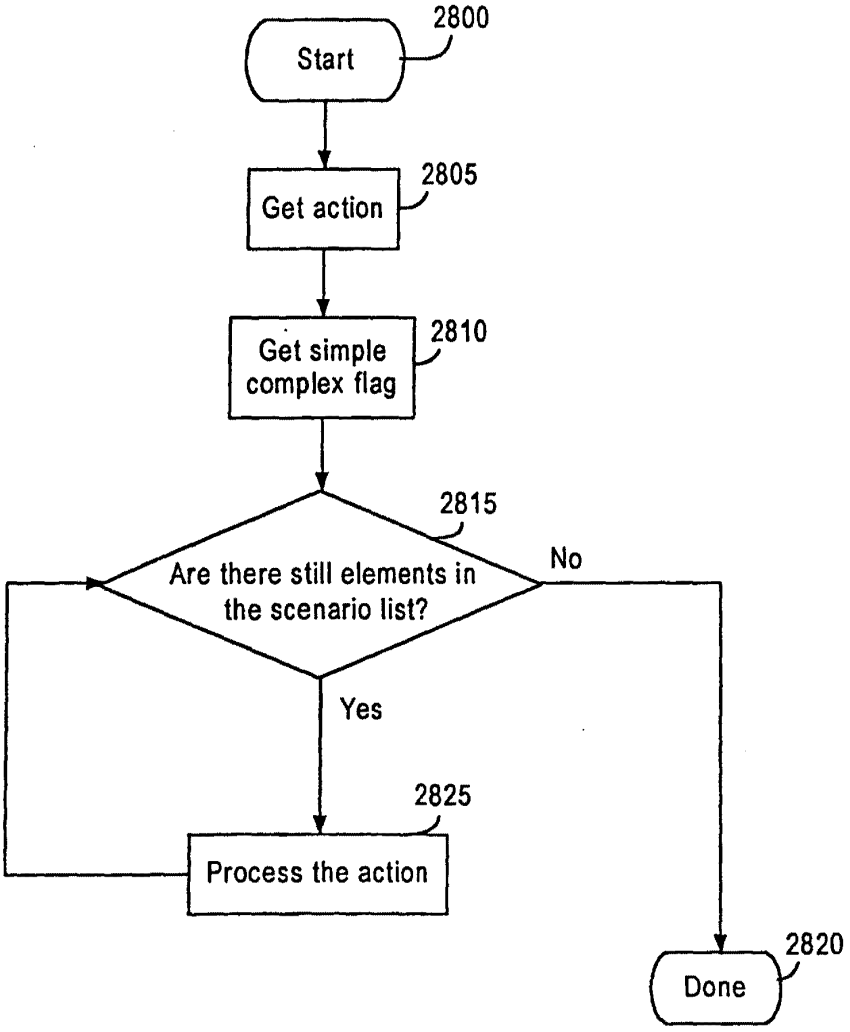

2800
Start
2805
Get action
2810
Get simple complex flag
2815
Are there still elements in the scenario list?
No
Yes
2825
Process the action
2820
Done

FIG. 29

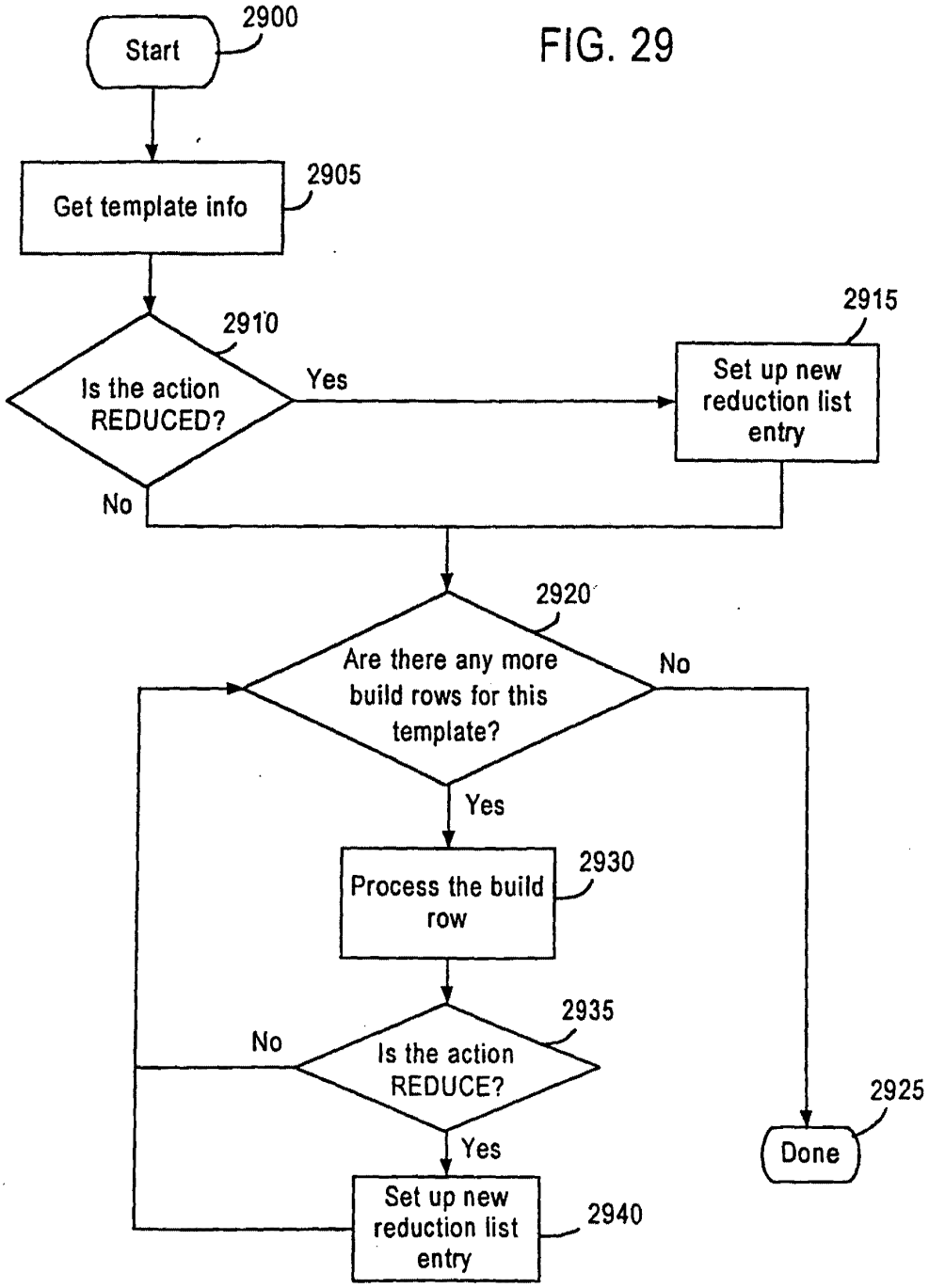

Start
2900
Get template info
2905
Is the action REDUCED?
2910
Yes
No
Set up new reduction list entry
2915
Are there any more build rows for this template?
2920
No
Yes
Process the build row
2930
Is the action REDUCE?
2935
No
Yes
Set up new reduction list entry
2940
Done
2925

FIG. 30

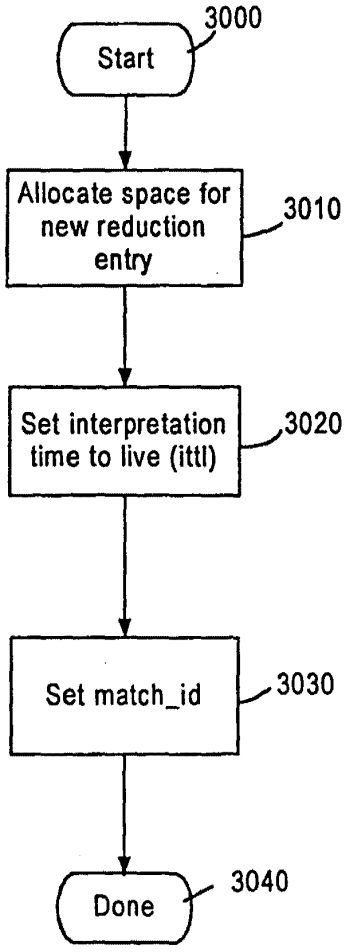

3000
Start
Allocate space for new reduction entry
3010
Set interpretation time to live (ittl)
3020
Set match_id
3030
Done
3040

FIG. 31

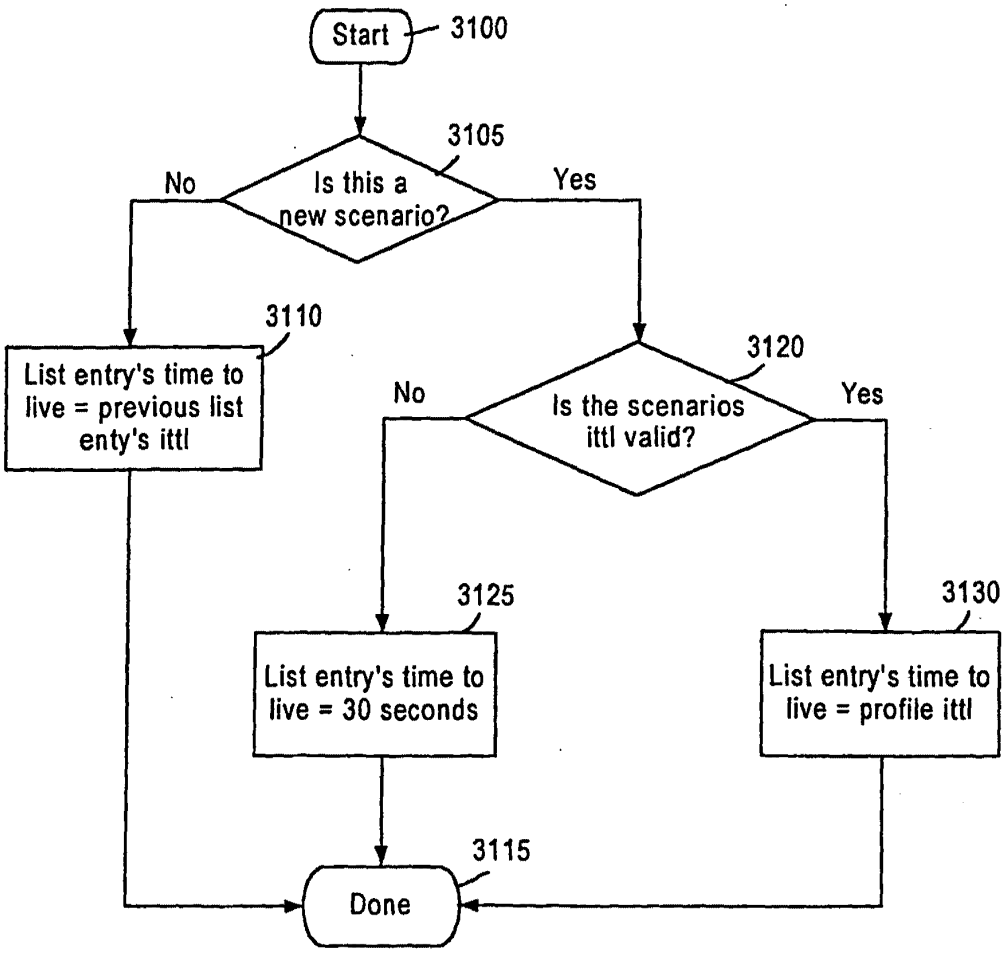
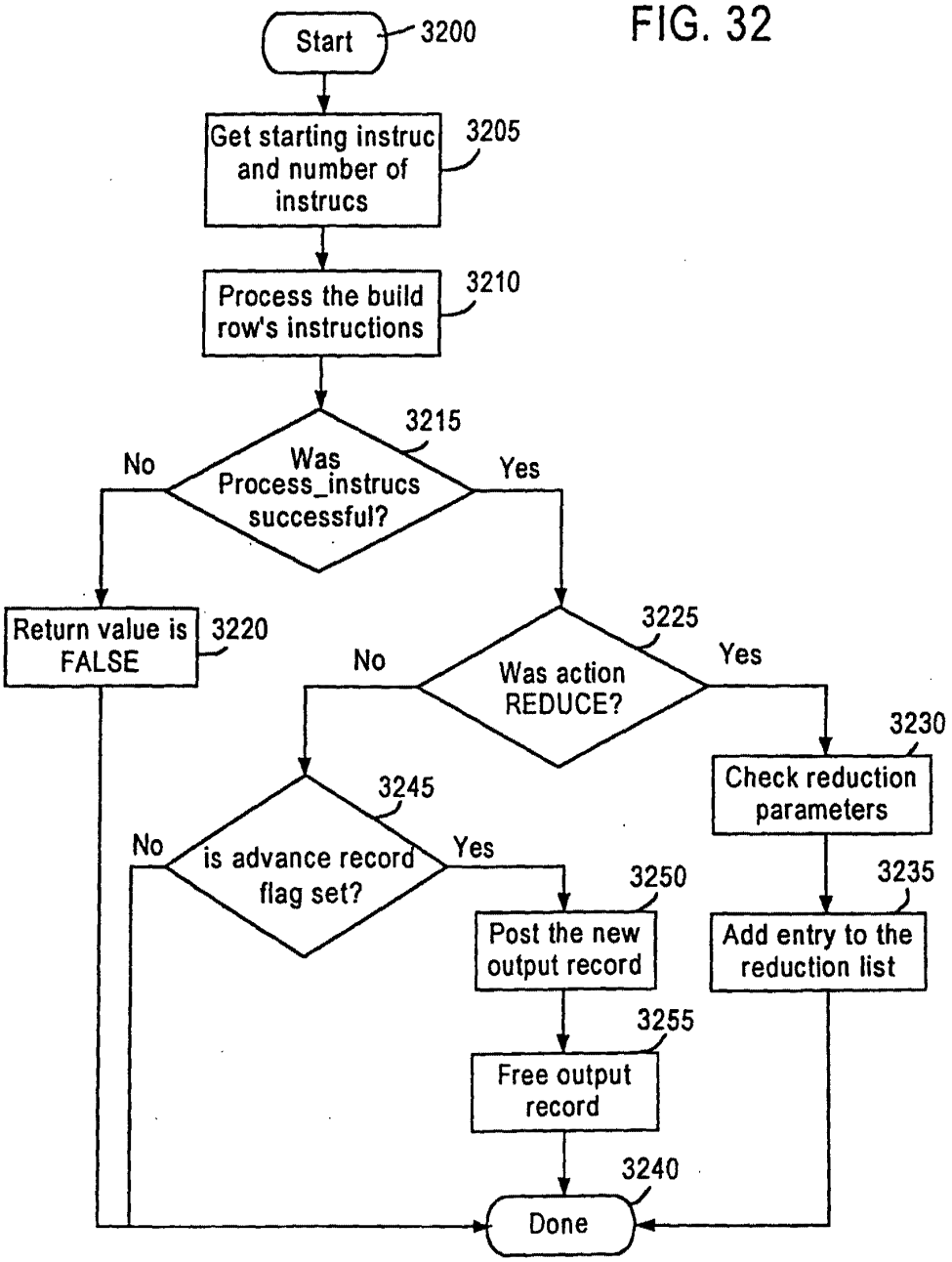

Start
3100
3105
Is this a new scenario?
No
Yes
3110
List entry's time to live = previous list enty's ittl
3120
Is the scenarios ittl valid?
No
Yes
3125
List entry's time to live = 30 seconds
3130
List entry's time to live = profile ittl
3115
Done

FIG. 32


Start
3200
Get starting instruc and number of instrucs
3205
Process the build row's instructions
3210
3215
Was Process_instrucs successful?
No
Yes
Return value is FALSE
3220
3225
Was action REDUCE?
No
Yes
3230
Check reduction parameters
3245
is advance record flag set?
No
Yes
3250
Post the new output record
3235
Add entry to the reduction list
3255
Free output record
3240
Done

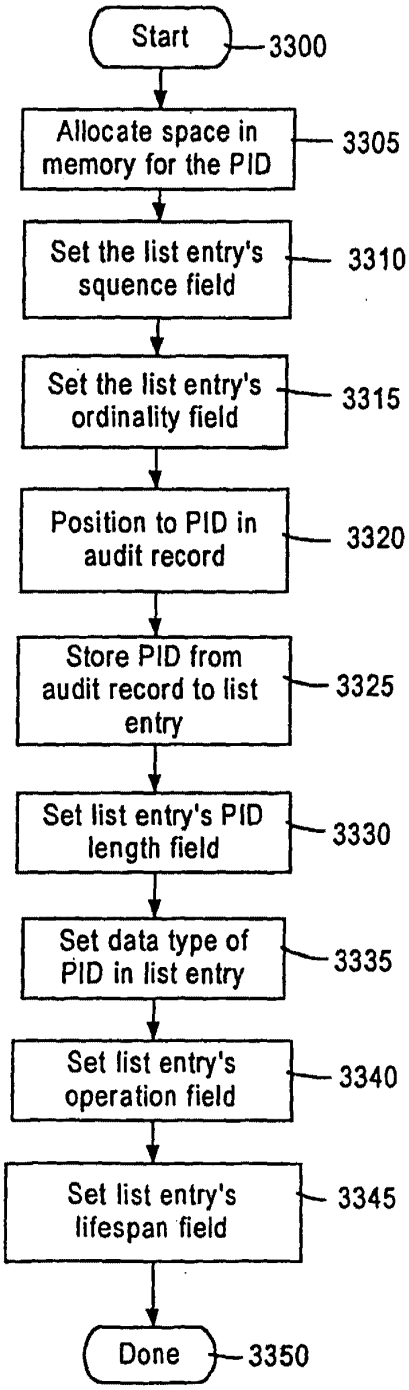

Start
3300
Allocate space in memory for the PID
3305
Set the list entry's squence field
3310
Set the list entry's ordinality field
3315
Position to PID in audit record
3320
Store PID from audit record to list entry
3325
Set list entry's PID length field
3330
Set data type of PID in list entry
3335
Set list entry's operation field
3340
Set list entry's lifespan field
3345
Done
3350

FIG. 33

FIG.34

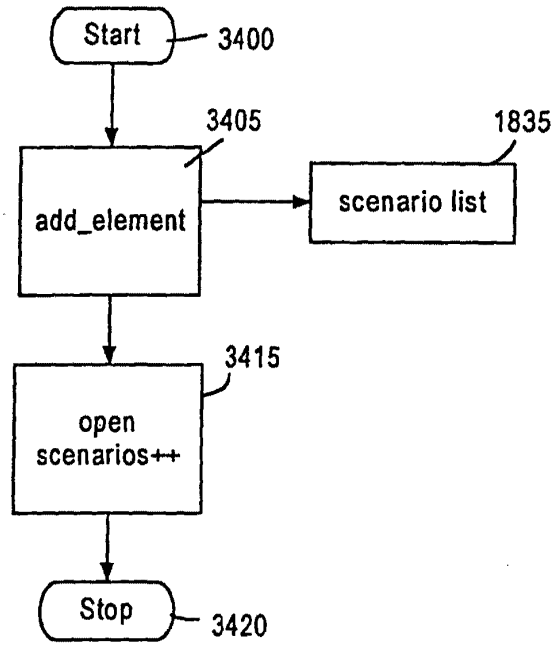

Start
3400
3405
1835
add_element
scenario list
3415
open
scenarios++
Stop
3420

FIG. 35

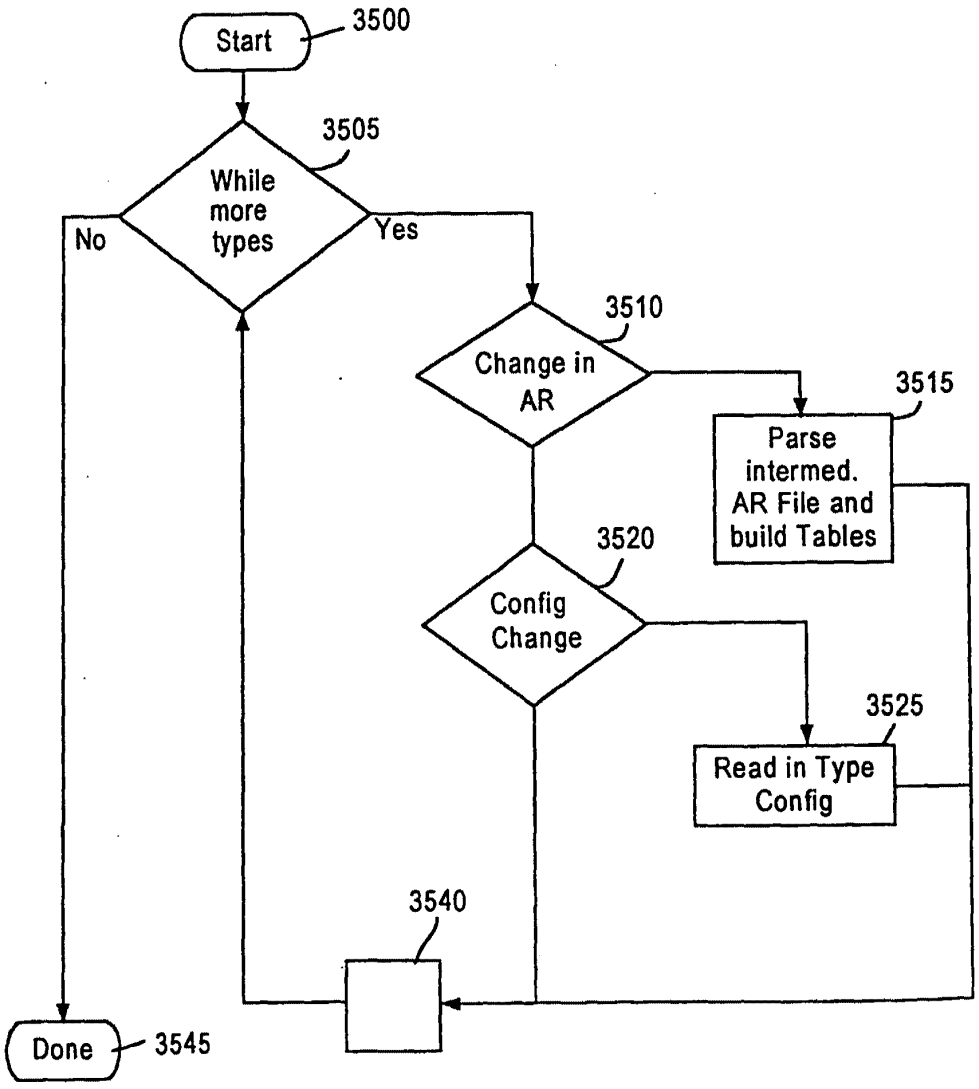

Start
3500
3505
While more types
No
Yes
3510
Change in AR
3515
Parse intermed. AR File and build Tables
3520
Config Change
3525
Read in Type Config
3540
Done
3545

FIG.36

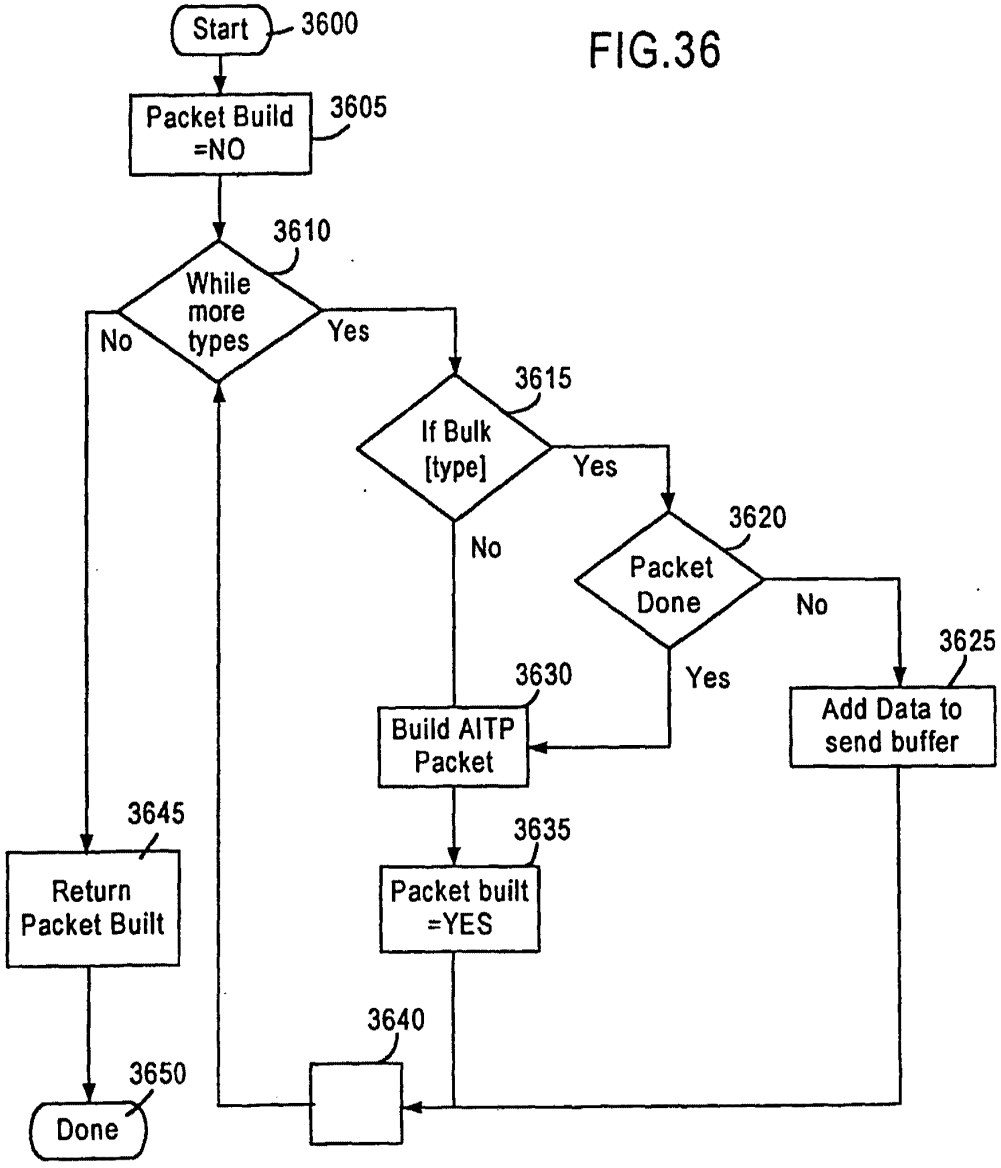

Start
3600
Packet Build =NO
3605
3610
While more types
No
Yes
3615
If Bulk [type]
Yes
No
3620
Packet Done
No
Yes
3625
Add Data to send buffer
3630
Build AITP Packet
3635
Packet built =YES
3640
3645
Return Packet Built
3650
Done

FIG.37

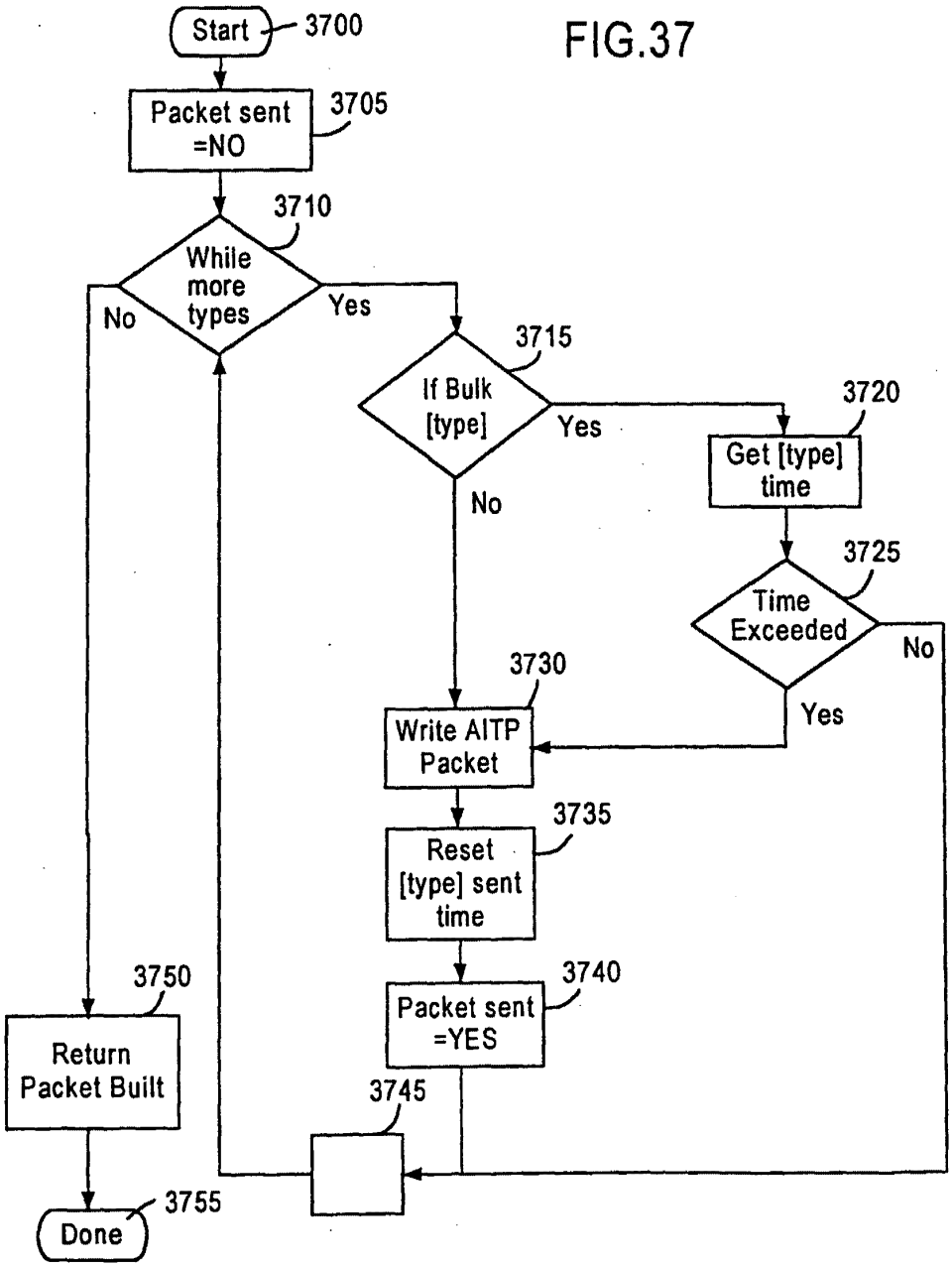

Start
3700
Packet sent =NO
3705
3710
While more types
No
Yes
3715
If Bulk [type]
Yes
No
3720
Get [type] time
3725
Time Exceeded
No
Yes
3730
Write AITP Packet
3735
Reset [type] sent time
3740
Packet sent =YES
3745
3750
Return Packet Built
3755
Done

FIG. 38

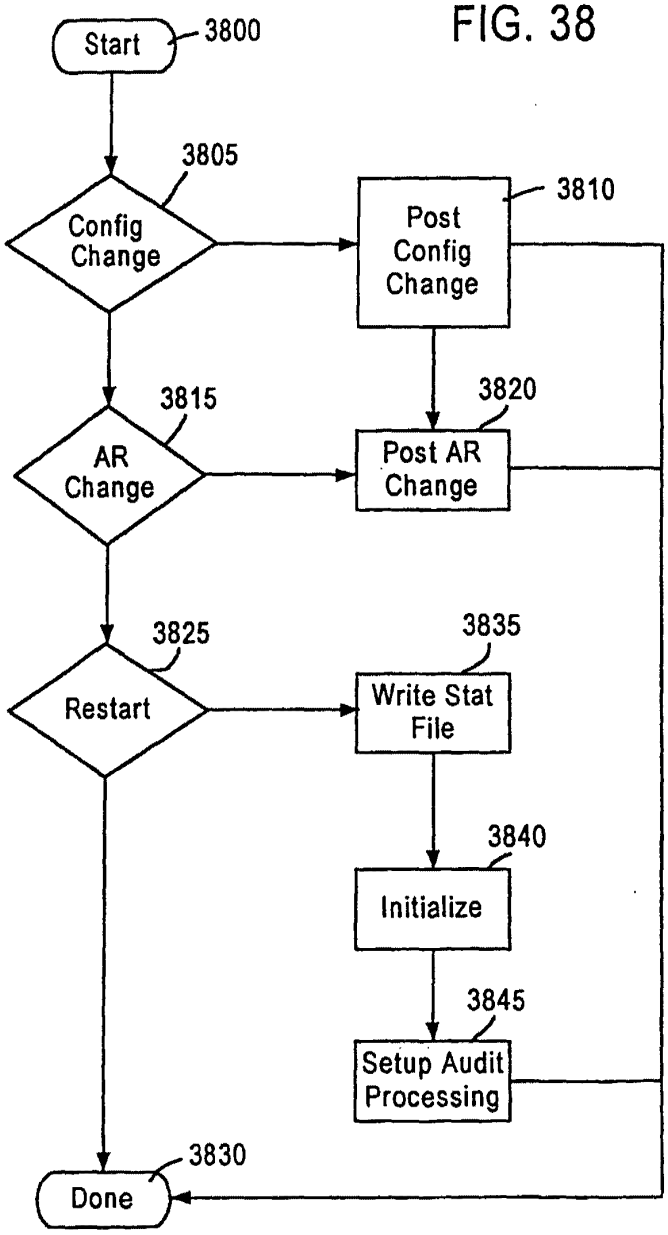
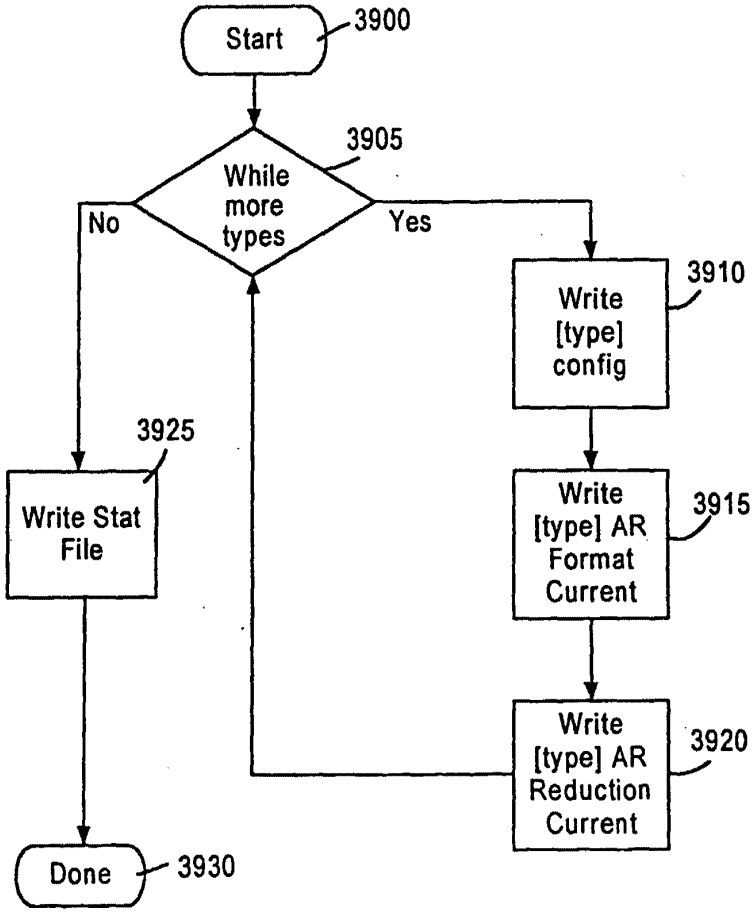

Start
3800
3805
Config Change
3810
Post Config Change
3815
AR Change
3820
Post AR Change
3825
Restart
3835
Write Stat File
3840
Initialize
3845
Setup Audit Processing
3830
Done

FIG. 39


Start
3900
While more types
3905
No
Yes
Write [type] config
3910
Write [type] AR Format Current
3915
Write [type] AR Reduction Current
3920
Write Stat File
3925
Done
3930

INTERNATIONAL SEARCH REPORT

Ir tional Application No
PCT/US 99/12181

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 745 753 A (MOSHER JR MALCOLM) 28 April 1998 (1998-04-28) figures 1-6 column 9, line 26 -column 10, line 57 column 11, line 43 -column 14, line 37	1-4,6,7, 15-17
A	US 5 561 795 A (SARKAR SHYAM S) 1 October 1996 (1996-10-01) figures 1,2,8,10,11,13 column 5, line 9 -column 7, line 30 column 9, line 35 - line 58 column 10, line 24 -column 13, line 10 column 13, line 50 -column 14, line 35	1-4,7, 10,11, 15-17

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 September 1999	06/10/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Weiss, P

1

Form PCT/ISA/210 (second sheet) (July 1992)

IPR2024-00027
CrowdStrike Exhibit 1002 Page 866

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 99/12181

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5745753 A	28-04-1998	CA 2167902 A	25-07-1996
		EP 0724223 A	31-07-1996
US 5561795 A	01-10-1996	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/00 — A2

(11) International Publication Number: **WO 99/00720**

(43) International Publication Date: 7 January 1999 (07.01.99)

(21) International Application Number: PCT/SE98/01257

(22) International Filing Date: 25 June 1998 (25.06.98)

(30) Priority Data:
9702476-4 26 June 1997 (26.06.97) SE

(71) Applicant *(for all designated States except US)*: TELEFONAKTIEBOLAGET LM ERICSSON [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: LARSSON, Martin [SE/SE]; Fackelrosgatan 26, S-653 45 Karlstad (SE).

(74) Agent: GÖTEBORGS PATENTBYRÅ; Sjöporten 4, S-417 64 Göteborg (SE).

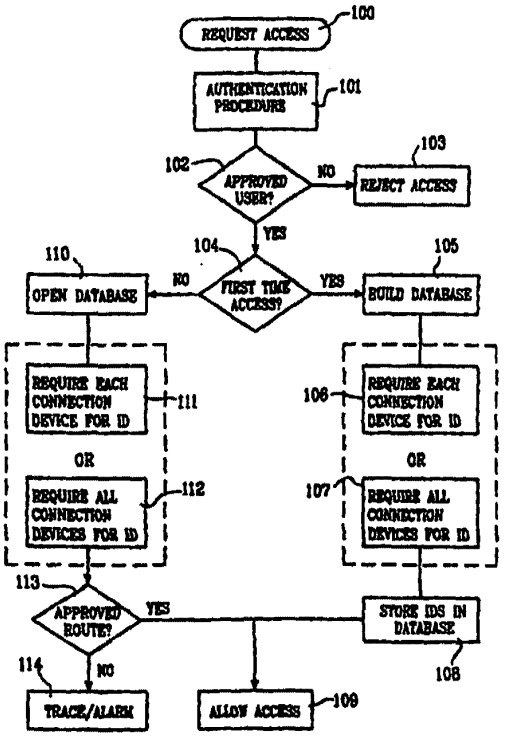
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD AND ARRANGEMENT RELATING TO COMMUNICATIONS SYSTEMS

(57) Abstract

The invention relates to a method and arrangement for detecting and/or tracking a non-authorised client station (11, 20) when accessing a host station (10, 21) in a communications network (12), which also comprises connecting devices (13-17, 22-28) in a route between said client station and host station (10, 21), each client station (11, 20), host station (10, 21) and at least some of said connecting devices (13-17, 22-28) being provided with a unique identity, the method comprising the steps of executing a first verification. The method in case of approved said first verification includes further step of route control comprising: retrieving the unique identity of each of said devices in said route, by propagating an identity inquiry, collecting an identity inquiry response message, including the identity of a least each device having a unique identity, comparing each unique identity of each device and/or station included in said response message with a list of approved identities, and rejecting or accepting the access, based on the comparison result.



IPR2024-00027
CrowdStrike Exhibit 1002 Page 868

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

TITLE

METHOD AND ARRANGEMENT RELATING TO COMMUNICATIONS SYSTEMS

TECHNICAL FIELD OF THE INVENTION

The present invention relates to a method and arrangement for detecting and preferably tracking a non-authorised client stations attempt to access a host in a communications network. The communication network comprises connection devices at least each having a unique identity.

The invention also relates to a telecommunications network having a security arrangement.

DESCRIPTION OF THE RELATED ART

During recent years, the art of digitalised communications using for example computers, mobile phones etc., has changed dramatically. The changes, besides of being an aid for users and allowing faster and better communication, have given some people opportunity to use the benefits and possibilities of essentially advanced communication means to carry out, more or less, criminal acts, such as fraudulence, e.g. by accessing the company or government computer networks and retrieving, changing or deleting information, or using telephones, switchboards etc. to obtain felonious privileges.

Internet, the global network for computers and computer networks have also contributed to the global communications, by allowing transference of images, voices and other data in a simple and inexpensive way. The result is that companies, government offices, universities and so on, have connected their networks through Internet to supply the internal and external users with relevant information and also to communicate with each other.

As the Internet is a public network, i.e. everybody having a computer and a communication device, such as a modem, may through an Internet Provider Server (IPS) access the Internet and communicate with others or just retrieve information. Internet is, in a superior manner, the fastest and most effective way to distribute a large amount of information for a large number of people.

CONFIRMATION COPY

IPR2024-00027
CrowdStrike Exhibit 1002 Page 870

The "core" of the Internet includes a very large number of computers, standalone or connected in networks, which can exchange information substantially directly using some predetermined protocols, especially Transmission Control Protocol/Internet Protocol (TCP/IP).

Each computer or connecting device in the core is separated from each other by means of an IP address. The IP address consists of a network number. In some cases the IP address is permanently assigned to a device (computer) in other cases the IP address is assigned to a computer temporarily. The IP address provides each connected computer/device a unique identity in the network.

The data transmission may be carried out via, for example fibre-optic lines, satellite links and telephone lines.

At present, obtaining full identification of a client or a user workstation connected or trying to connect to a host system is not possible. Also, It is not possible, in a simple and fast way, to identify the fraud workstation and thereby the user, even though a partial identification is possible.

When accessing a network, usually a login procedure is executed for authentication of the user. The authentication works, by the client first declaring the user name to be used to access the network. The service providing server then responds with a set of authentication methods, which are acceptable. The client then sends an authentication request, and this dialogue continues until an access has been granted or denied. The authentication methods can vary from system to system. Some methods are:

none	checks if no authentication is ok,
password	a conventional password authentication, which requires a password for access,
secureid	secureID authentication is a timing-based hardware token identification, where the user enters a code displayed on the token as authentication.

Also, on-time passwords and similar methods are available. As other methods public key can be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 871

mentioned, in which the possession of a privet key is the authentication.

Great efforts are made to develop methods and algorithms to secure the authentication procedures, but no system is more secure than the user of the system, as passwords and keys may come in possession of non authorised persons. In the Internet case, also the IP addresses can be forged which allows accessing the network without problem.

By forged IP address, it is also possible to attack the Internet service providers, for example by flooding or the like. Flooding is a method where an unreachable source (IP) address is used against a target host computer, which attempts to reserve resources waiting for a response. The attacker repeatedly changes the bogus source address on each new access packet sent, thus exhausting additional host resources. Then, if the attacker uses some valid address as the source address the attacked system responds by sending a large number of reply packages, which at end, results in a degraded performance and even system crash.

U.S. Patent no. 5,619,657 teaches a method for providing security facility for a network of management servers utilising a database of trust relations to verify mutual relations between management servers. The method relates specially to creating accounts on a system over a network. The management operation (MO) also contains the identity of the user lunching the operation. Through an interface the MO is transferred to a dispatcher of the management server (MS). The MS, in addition to administrating, requests for management services provided by a local system is also responsible for routing MOs on secure paths to other local systems in the network and managing the security of the local system. The MS determines a proper link by means of a database, which maintains trusted relations between the management servers. The trusted relation lists are generated independent from the execution of a communication protocol by an autonomous network utility. Each MS contains a list. The lists are divided into two categories, trusted receivers and trusted senders. Based on the trusted list, forwarding the operations are executed. In summary, the database provides a means for routing MOs from one MS to another MS along a secure path determined by the trust relations of the MSs at each link in the route in the network performing the MO.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 872

SUMMARY

There is a need for a method and arrangement in communications system, specially in a computer network, which provides a simple, effective and straightforward way to increase the security.

There is also a need for a method and arrangement, which allow to detect an unauthorized access attempt to a service providing device or the like in a communications system by a user station or process, by using substantially available procedures.

Through implementation of the method and arrangement, according to the invention there is provided a fast way to trace the position of the fraud user stations.

Therefore the method according to the invention, in case of a first approved control of the identity of a client station trying to access the network, further comprises steps of: retrieving the unique identity of each of said devices in said route, by propagating an identity inquiry, collecting an identity inquiry response message, including the identity of at least each device having a unique identity, sent by each device having an identity and arranged to response, comparing each unique identity of each device and/or station included in said response message with a list of approved identities, and rejecting or accepting the access, based on the comparison result.

In an advantageous embodiment the first verification includes controlling the client station identity and comparing it with a list of approved identities and/or number of accesses made by the client station. If the client identity approves and a first time access is detected the method further comprises the steps of: retrieving the unique identity of each of said devices in said route, arranging a database for said client station for storing the connection route by, and including the identity of each device having a unique identity sent by each device having an identity and arranged to response in the database.

It is possible that the identity of each device is collected by propagating an identity requirement and collecting the identity requirement response message. Then a list including said identities received in said response message is arranged, where the identity list constitutes a route list and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 873

the list is compared with a list of an approved route list.

In an embodiment, the comparison of the client station identity with a list of known approved clients, is carried out before the route control, which is possible to carry out after the route control.

In the advantageous embodiment, the communications network is a computer network and the unique identity is IP address of the client station, host station and at least some of said connecting devices. Preferably, the communications network is Internet. The communications network may also be Interanet/Extranet.

In another embodiment, an alarm signal is generated in case of rejection or a trace procedure in is executed.

In an embodiment an error message inquiry is sent to each or all devices, asking for identity conflict errors and/or other errors.

According to the invention a security arrangement is provided for detecting and/or tracking a non-authorised user access to a communications network. The network comprises at least one client station at least one host station and connecting devices in a route between said client station and host station, each client station, host station and at least some of said connecting devices being provided with a unique identity. The arrangement comprises means for communication with said host station, client station and connecting devices, memory units for storing information, means for fetching the identity of the client station at login, a comparison device for comparing the identity of the user with a stored identity list, means to propagate a device identity inquiry, means to identify and collect responses to said inquiry including unique identities and means to compare said collected identities with a stored list of identities.

Advantageously, the arrangement is integrated in said host station or separately arranged in a supervising server. The connection devices may be any one of the routers; infrastructure devices such as bridges, terminal servers, gateways, firewalls, repeaters, application servers such as DNS (Domain Name System), mailhubs, news servers, FTP (File Transfer Protocol)

IPR2024-00027
CrowdStrike Exhibit 1002 Page 874

servers, WWW (World Wide Webb) servers, network management systems and communication servers.

A telecommunications network according to present invention, including a calling unit, one or more switching stations and links and a receiving station is characterised by a security arrangement for detecting and/or tracking a non-authorised user when connecting speech for said calling unit, each of the calling unit, the switching stations/links being provided with a unique identity. The arrangement further comprising means for collecting identity of each calling unit, switching station, upon a request from said arrangement and creating a list for a call route, comparing said call route with approved call routes and approving or rejecting the call.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be further described in the following in a non-limiting way under reference to the enclosed drawings in which:

Fig. 1 is a schematic illustration of a first embodiment of a system implementing the invention.

Fig. 2 is a flowchart, schematically illustrating the function of the embodiment according to Fig. 1.

Fig. 3 illustrates another embodiment of the invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In the following, the invention will be disclosed referring to two non limiting embodiments. The first embodiment is based on a computer network using IP addresses (TCP/IP protocol), examples of such networks are Internet, intranet/Extranet. The second embodiment is a telecommunications network.

It is assumed that the TCP/IP protocol and IP address structure is known for a skilled person, but briefly its operation is based on dividing a message into small packages or datagarms, each of which are transmitted individually and collected to the original message at the destination set. This makes it possible for the packages to take a free route if a route is busy.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 875

Fig. 1 illustrates a computer network including at least one service providing server (SPS) or supervising server 10. A user or client station (CS) 11 can access the SPS 10 through a network 12 including one or several connection devices 13 - 17. The SPS 10, CS 11 and substantially each connection device is provided with a unique identity, such as an IP address. Furthermore, the SPS 10 is arranged with a storing means 18. The CS 11 may also be a member of another local network, connected to other computers in the local network by means of network cards in a known manner. In this case the connection device 13 may be a router or the like.

The connection devices and devices having IP address may be any of the:

* Routers;
* Infrastructure devices such as bridges, terminal servers, gateways, firewalls, repeaters;
* Application servers such as DNS (Domain Name System), mailhubs, news servers, FTP (File Transfer Protocol) servers, WWW (World Wide Webb) servers, network management systems etc.;
* Peripheral devices, such as printers, printer servers, CD-ROM servers, communication servers etc.

Generally, above-mentioned devices are arranged to return their unique address if a query is directed at them.

The basic idea of the invention will be clear through the following non-limiting operational example with reference to the flowchart of fig. 2.

The client station 11 directs 100 an access request to one or several servers 10, which also can be a security management server. An authentication procedure 101 is then launched by the server 10 requiring a password or the like. The reply from the client station is then compared with, 102, a list of approved identities stored in the database, for example stored in device 18 of the server 10. If the user name and password are in the list the first step of access is approved, otherwise the access is rejected 103 in a known way. The server 10 may only check the IP address of the CS 11, and use the IP address for further authentication process.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 876

If the first step of access is approved, the authentication procedure checks 104, if it is the first time for a client station accessing the server (system). If it is first time access, a new database for the client is built 105. Then the procedure propagates a query on the network, directed to each 106 device 13-17 or all 107 devices 13-17 in the route, through which the communication between the server 10 and the client station 11 is established. The query asks for the identity (IP address) of each connection device 13-17 in the route. If a device is provided with an identity, at receiving the identity query, it transmits back the identity. The identities received by the server are then collected and stored 108 in the database. Then, as this being the first access, the client station is allowed to access 109 the network (services).

At step 104, if it is dedicated that it is not the first access of the client station, a database including data about the client station or dedicated to the client station, including for instance the identities of the connection devices collected by means of a procedure according to previous section, is opened. Then, the procedure propagates a query on the network, directed to each 111 device 13-17 or all 112 devices 13-17 in the route, through which the communication between the server 10 and the client station 11 is established. The query asks for the identity (IP address) of each connection device in the route. If a device is provided with an identity, at receiving the identity query, it transmits back its identity. The identities received by the server are then collected and compared 113 with the list of identities in the database. If the lists of identities confirm, the access is accepted 109. Otherwise, an alarm is generated 114. As the identity of each connection device corresponding to a physical site is retrieved, it is possible to trace 114 the physical location of the client station, even though the user name, password and the IP address are forged.

Moreover, it is possible to control attempts to build a false rout or change the IP addresses without permission. The arrangement can transmit a query, asking the connection devices for any identity conflicts or any error messages related to identity problems. If an identity conflict or error message is detected, then the system may report the detection to a system operator and ask for a manual sanction.

Controlling the first time access is, for example carried out by inspecting a corresponding database for existence of the client station identity (e.g. user name or IP address). In some

IPR2024-00027
CrowdStrike Exhibit 1002 Page 877

cases, the identities in the route can vary, for example when a connection device having permanent identity is replaced or the message takes another route. In these cases, it is possible to add the new identity route into the database, preferably after a double check that the client station is operated by an authorized client. It is also possible to arrange the connection devices to report the replacement to a controlling database, whereby the server can check for possible changes. Experiments have shown that the datagrams in a TCP/IP based network, substantially take always the same route.

It is also possible to execute the first authentication procedure 102, asking for user-name and password, after step of route approvals 108 or 113.

The queries for identities of the devices may be executed using existing procedures or special procedures. It is possible to send an "echo request" in form of a data packet to a remote host and wait for each echo reply from each device. Using a traceroute function the server can trace and register the actual route that an IP packet follows to a network, Internet or intranet host. The function "finger" returns the users logged on to remote systems. It is also possible to send a message which returns the entry from a remote registry and information about domains (WHOIS). It is also possible to use net scanning functions, preferably over a specified range of IP addresses, for example by pinging each one, which also returns the names and adding them to the possible host files.

Using the premises of the invention, to avoid a flooding attack, a server may control the route to the origin of the (forged) IP address and take a proper action if needed, and thereby avoid unnecessary ACK/REQ transmissions, which degrades the performance of the system.

It is also possible to arrange the devices not having IP addresses with other unique identities, which can be stored in the route database of the server.

Fig. 3 shows a telecommunications system employing the method according to the invention. The system includes a calling unit 20, a receiving unit 21 and connecting devices, which can be one or several of base stations 22, switch boards 23, 28, links 24, satellite relays 25, 27 and satellites 26. Preferably, each of the devices 20-28 are provided with a unique identity. When a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 878

call is switched, preferably the base station 28 or the receiving unit 21 is arranged with a database including the call route for a call from each authorised calling unit 20. Particularly, in a digitalised network, each device can be arranged to, automatically or on request, add its identity to a message to the base station 28 or the receiving unit 21. Using the received list of the devices in the route, the base station 28 or the receiving unit 21 can decide to allow or not allow the call. This embodiment may be applied to any telecommunications system, components of which have or can be provided with identities, such as GSM, NMT etc.

This system could particularly be useful, for example for security calls, banking transactions etc.

The invention is not limited to the illustrated embodiments but can be varied in a number of ways without departing from the scope of the appended claims and the arrangement and the method can be implemented in various ways depending on application, functional units, needs and requirements etc.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 879

CLAIMS

1. A method for detecting and/or tracking a non-authorised client station (11, 20) when accessing a host station (10, 21) in a communications network (12), which also comprises connecting devices (13-17, 22-28) in a route between said client station and host station (10, 21), each client station (11, 20), host station (10, 21) and at least some of said connecting devices (13-17, 22-28) being provided with a unique identity, the method comprising the steps of executing a first verification,

characterised in,

that the method, in case of approved said first verification includes further step of route control comprising:

- retrieving the unique identity of each of said devices in said route, by propagating an identity inquiry,
- collecting an identity inquiry response message, including the identity of at least each device having a unique identity,
- comparing each unique identity of each device and/or station included in said response message with a list of approved identities, and
- rejecting or accepting the access, based on the comparison result.

2. The method according to claim 1,

characterised in,

that said first verification includes controlling the client station (11, 20) identity and comparing it with a list of approved identities.

3. The method according to claim 1 or 2,

characterised in,

that said first verification includes controlling number of accesses made by the client station (11, 20).

4. The method according to claim 3,

characterised in,

that if the client identity approves and a first time access is detected the method further comprises the steps of:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 880

- retrieving the unique identity of each of said devices (13-17, 22-28) in said route
- arranging a database for said client station for storing the connection route by, and
- including the identity of each device (13-17, 22-28) having a unique identity sent by each device having an identity and arranged to response in the database.

5. The method according to claim 4,
characterised in,
that the identity of each device is collected by propagating an identity requirement and collecting the identity requirement response message.

6. The method according to claim 4,
characterised by,
arranging a list including said identities received in said response message, said identity list constituting a route list and comparing said list with a list of approved route list.

7. The method according to claim 2,
characterised in,
that said comparison of the client station identity by a list of known approved clients, is carried out before the route control.

8. The method according to claim 2,
characterised in,
that said comparison of the client station identity by a list of known approved clients, is carried out after the route control.

9. The method according any one of claims 1 - 8,
characterised in,
that said communications network is a computer network (12).

10. The method according claim 9,
characterised in,
that said unique identity is IP address of the client station (11), host station (11) and at least

IPR2024-00027
CrowdStrike Exhibit 1002 Page 881

some of said connecting devices (13-18).

11. The method according to claim 9 or 10,
characterised in,
that said communications network is Internet.

12. The method according to claim 9 or 10,
characterised in,
that said communications network is an Interanet/Extranet.

13. The method according to any one of claims 1-12,
characterised by,
generating an alarm signal in case of rejection.

14. The method according to any one of claims 1-13,
characterised by,
executing a trace procedure in case of rejection.

15. The method according to claim 1,
characterised in,
that an error message inquiry is send to each or all devices, asking for identity conflict errors and/or other errors.

16. The method according to claim 1,
characterised by,
denying access in case of a failure first verification.

17. A security arrangement for detecting and/or tracking a non-authorised user access to a communications network (12), which comprises at least one client station (11), at least one host station (10) and connecting devices (13-17) in a route between said client station and host station, each client station, host station and at least some of said connecting devices being provided with a unique identity,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 882

characterised in,

that the arrangement comprises means for communication between said host station, client station and connecting devices, memory units for storing information and instructions, means for fetching the identity of the client station at login, means for comparing the identity of the user with a stored identity list, means to propagate a device identity inquiry, means to identify and collect responses to said inquiry including unique identities and means to compare said collected identities with a stored list of identities.

18. The arrangement according to claim 17,

characterised in,

that the arrangement is integrated in said host station (10).

19. The arrangement according to claim 17,

characterised in,

that the arrangement is separately arranged in a supervising server.

20. The arrangement according to any one of claims 17 - 19,

characterised in,

that the route list is automatically updated.

21. The arrangement according to any one of claims 17 - 20,

characterised in,

that the connection devices are any one of routers; infrastructure devices such as bridges, terminal servers, gateways, firewalls, repeaters, application servers such as DNS (Domain Name System), mailhubs, news servers, FTP (File Transfer Protocol) servers, WWW (World Wide Webb) servers, network management systems and communication servers.

22. A telecommunications network including a calling unit (20), one or more switching stations and links (22-28) and a receiving station (21),

characterised by,

a security arrangement for detecting and/or tracking a non-authorised user when connecting speech for said calling unit (20), each of the calling unit (20), the switching stations/links (22-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 883

28) being provided with a unique identity,

said arrangement further comprising means for collecting identity of each calling unit (20), switching station (22-28), upon a request from said arrangement and creating a list for a call route, comparing said call route with approved call routes and approving or rejecting the call.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 884

FIG.1

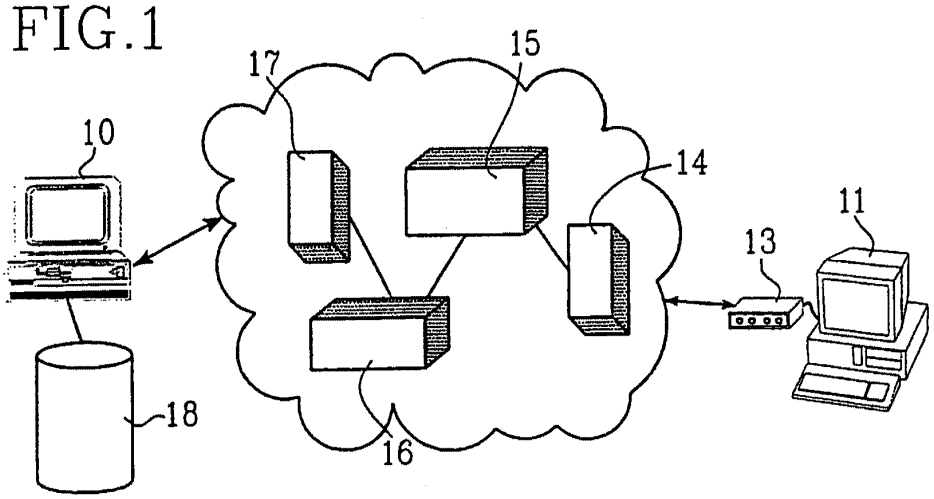


FIG.3

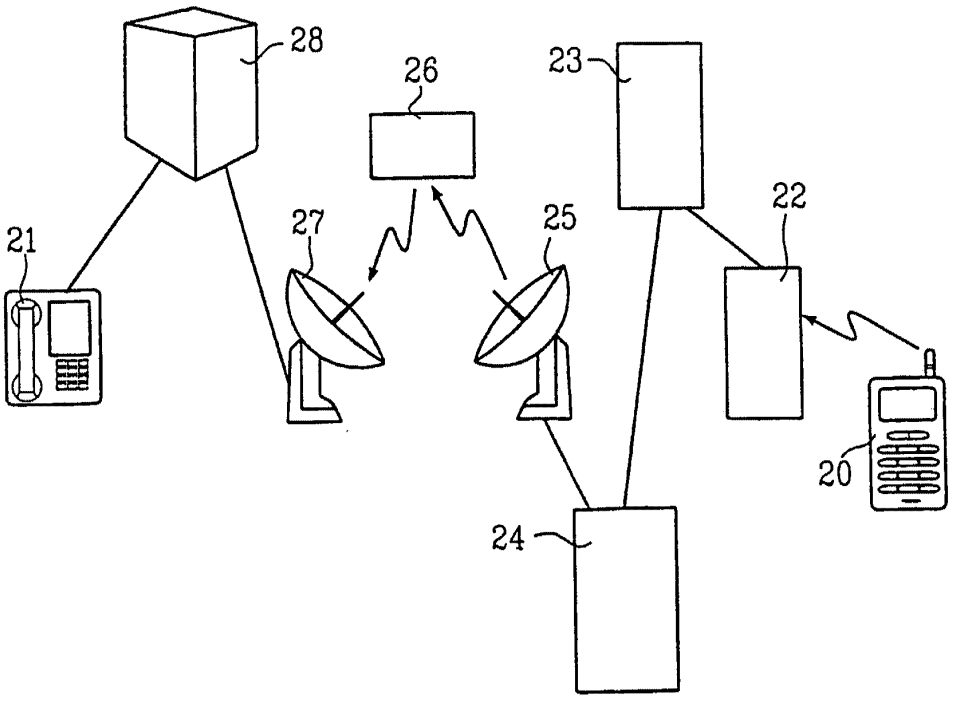
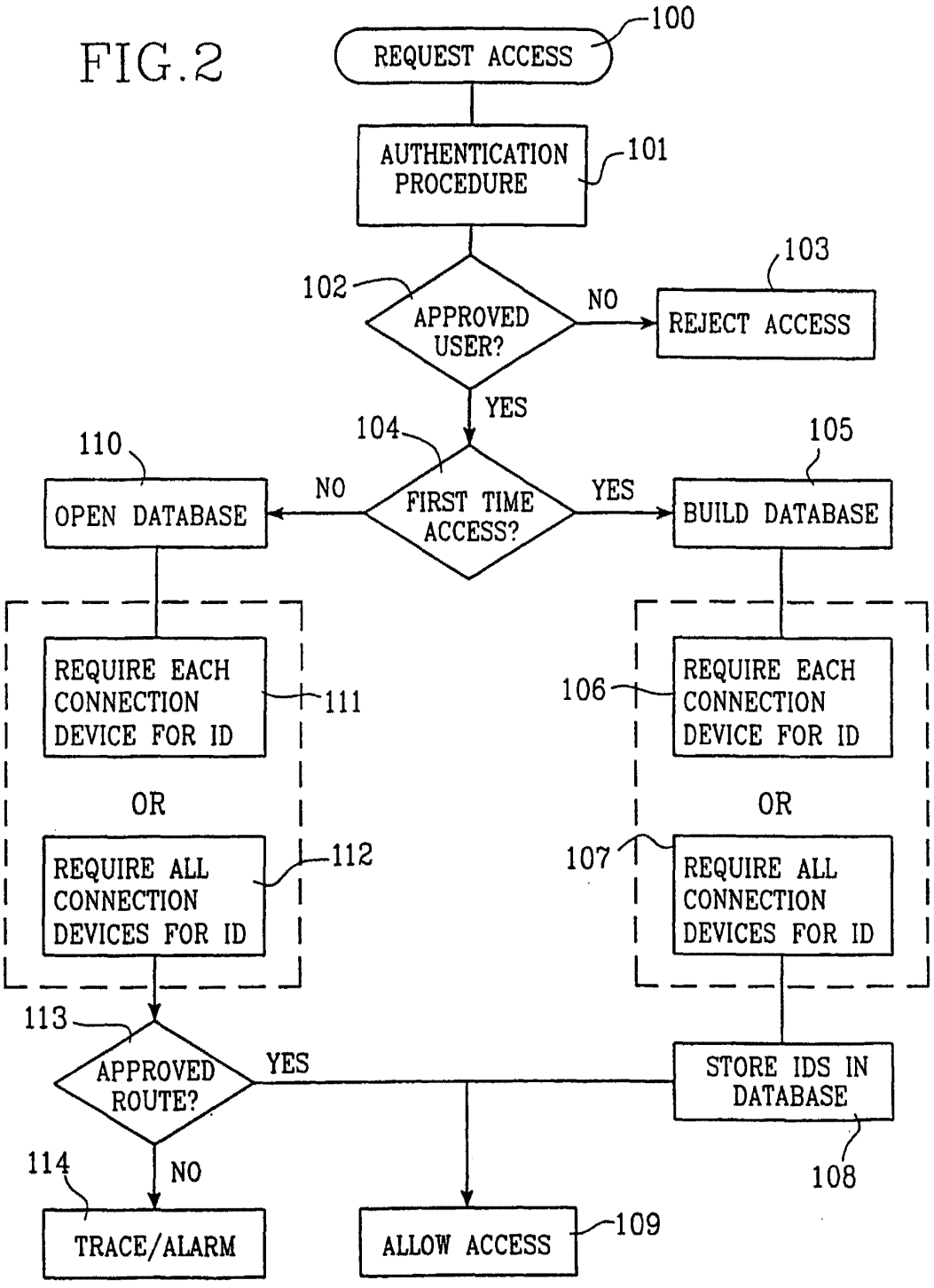


FIG.2



IPR2024-00027
CrowdStrike Exhibit 1002 Page 886

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/00 — A1

(11) International Publication Number: **WO 99/50734**

(43) International Publication Date: 7 October 1999 (07.10.99)

(21) International Application Number: PCT/US99/05025

(22) International Filing Date: 8 March 1999 (08.03.99)

(30) Priority Data:
09/052,844 31 March 1998 (31.03.98) US

(71) Applicant: AT & T CORP. [US/US]; 32 Avenue of the Americas, New York, NY 10013-2412 (US).

(72) Inventors: GOLDBERG, Randy, G.; 48 Primrose Circle, Princeton, NJ 08540 (US). ROSEN, Kenneth, H.; 107 Red Hill Road, Middletown, NJ 07748 (US). SALIMANDO, Steven, Charles; 22 N. Rivers Edge Drive, Little Silver, NJ 07739 (US).

(74) Agents: DWORETSKY, Samuel, H. et al.; AT & T CORP., P.O. Box 4110, Middletown, NJ 07748 (US).

(81) Designated States: BR, CA, CN, JP, MX, NO, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: A METHOD OF AND APPARATUS FOR COMPUTER SECURITY USING A TRANSMITTING LOCATION DEVICE

(57) Abstract

A method and apparatus enhance computer security based on pre-registration and tracking of a computer user's location. A location device accompanies an individual attempting to log-in to a computer network from a location distant to the network. When the location device is activated, it transmits a location signal which is received by the computer network. The network then determines from the location signal where the individual is located during the log-in process. If the individual is at a pre-defined physical location, the computer grants access; otherwise, access is denied.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 887

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A METHOD OF AND APPARATUS FOR COMPUTER SECURITY USING A TRANSMITTING LOCATION DEVICE

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to a method of and apparatus for computer security using a transmitting location device, and, more particularly, to a method of and apparatus for adding an additional layer of computer security based on registration and tracking of the computer user's location.

DESCRIPTION OF RELATED ART

The increasing use of remote access, that is the use of a computer or other device to communicate with a computer network from a location distant from the network, has enabled individuals who otherwise do not have authorized access to the computer network to none-the-less violate computer security from afar. The ability of computer hackers to infiltrate a computer network from a distant location can be a serious threat to a company's well-being. This threat is especially serious for companies which have an ever increasing reliance on a workforce who tele-commute to work from home everyday. Thus knowing where an individual is when they attempt to gain entry to the computer network would be an important aspect of computer security.

Position detection for locating individuals, devices, and vehicles has been accomplished. For example, U.S. Patent No. 5,689,269, issued November 18, 1997 to Norris, relates to an apparatus and method for determining the position of a first device relative to the position of a second device using the Global Positioning System (GPS). The first device, with a person or object to be located, transmits telemetry position data to the second device after first receiving a GPS signal and determining its own location using that GPS signal. The second device receives

IPR2024-00027
CrowdStrike Exhibit 1002 Page 889

the telemetry position data from the first device and calculates a relative distance between the two devices. The calculation performed by the second device is based on the telemetry position data received from the first device and knowledge about its own position determined from GPS signals that it has previously received. The second device is also capable of determining direction and difference in elevation between the first and second devices.

Further, U.S. Patent No. 5,550,551, issued August 27, 1996 to Alesio, relates to a position monitoring system and method particularly applicable to vehicle monitoring. When activated, a position detector mounted on the vehicle uses GPS signals to determine vehicle location information. On a pre-determined basis, the position detector periodically updates the vehicle location information and transmits a location information signal based on the vehicle's location to a remote dispatch center. The dispatch center receives the transmitted location information signal from the position detector, determines the vehicle location, and relays that information to an appropriate law enforcement agency.

Yet another example, U.S. Patent No. 5,389,934, issued February 14, 1995 to Kass, relates to a portable system for locating a person, vehicle or object. The system uses a GPS unit and a piece of cellular telephone equipment. The system's locating function is first activated by receipt of a telephone call on the piece of cellular telephone equipment. Upon this activation, the system then determines its own location via the GPS unit and responds to the call with a voice message stating its current location. The person, vehicle or object may then be retrieved.

As can be seen, however, while the ability to accurately locate a person, device or vehicle exists, this ability has not been applied to help with computer security. As the threat of a breach of computer security from afar still exists, and in fact seems to be increasing, there still remains a need for a method of enhancing computer security based on detection of location.

SUMMARY OF THE INVENTION

Accordingly, in response the present invention, as embodied and broadly described herein, provides a method of and apparatus for adding an

IPR2024-00027
CrowdStrike Exhibit 1002 Page 890

additional layer of security to the computer log-in process based on registration and detection of location. Thus, an individual who wishes to log-in to a computer system must not only be an authorized user of the system, but must also be attempting to log-in from a pre-registered and authorized location or zone.

Proper location is checked through the use of a transmitting location device. When an individual who is an authorized user of a computer network desires to access that network from a location distant to the network, a location device is activated. Once activated, the location device will transmit a location signal to the computer network.

An additional layer of security is thus added through the use of the transmitting location device. This is accomplished when the individual is logging-in to the network. Once the locating device has been activated and is transmitting a locating signal, the computer network will receive that locating signal and determine where the individual is as they are attempting to log-in. The computer network will then match that determined location against a list of pre-registered locations. If the individual is in fact located at a location that has been pre-registered, the computer network will allow access using both the location information and the standard security measures (e.g. ID and password). Thus not only must the person be an authorized user (which can be determined by the ID and passcode, inter alia), but the location must be a pre-authorized and pre-registered location.

Further, the additional security may be added to the on-going session as well. As the individual is logged-on to the network, the network may reactivate the location device to periodically check the individual's location. Periodic updates allow the computer network to ensure that the individual is still at and/or in the pre-registered location or zone, and that a proper location signal is being received.

The present invention, including its features and advantages, will become more apparent from the following detailed description with reference to the accompanying drawings.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 891

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a flow chart of a method of enhancing computer security using a transmitting location device in which the location device transmits a location signal during an attempt to log-in to a secure computer network, according to an embodiment of the present invention.

Figure 2 illustrates a schematic of an apparatus by which transmission of the location signal from the location device to the computer network can be carried out, according to an embodiment of the present invention.

DETAILED DESCRIPTION

Figures 1 and 2 show a method and an apparatus for adding an additional layer of security to a computer log-in process based upon a pre-registration operation and a subsequent detection of a computer user's location. Thus the location from which the computer user attempts to log-in, and from which he or she continues to work, becomes an additional element by which computer security may be maintained. If the individual using the computer logs-in from a pre-registered location, and the central computer recognizes that location as an authorized location, log-in to the computer network is permitted. However, if the identified location is determined not to be an authorized location, log-in is not permitted. Subsequent updates of the computer user's location can also be used to ensure that the user is still in the authorized location. Thus if the individual is subsequently determined to be outside of the pre-registered location, access to the computer network can be terminated.

An individual who will have need of logging-in to a computer network from a location outside of the immediate area of the computer network will be required to pre-register the location (or locations) from which he or she shall be logging-in. A central computer will then consider each pre-registered location as an authorized location for that individual. In essence, then, the location is keyed to that individual and is the only location from which the individual may log-in and continue to work. Approval of the location by the central computer may be dependent upon any number of pre-set criteria. Further, ultimate approval may reside with the appropriate company personnel. It is to be understood, then, that

IPR2024-00027
CrowdStrike Exhibit 1002 Page 892

the location approval process may be established and administered in any manner which the company (and/or individuals) using the present invention approves. It is not to be limited to simply the embodiment herein described, and is only of importance in ensuring that the registered locations are in fact pre-approved.

The actual locations being pre-registered by the computer user may be a single place or a broader area. For instance, an individual may want to pre-register his or her home, and may also want to pre-register the area which follows a route to and from work. Such a registration scheme thus allows the individual to work from home and also to work while en route to or from work. It is to be understood, of course, that the number of locations which each individual may be allowed to register can be pre-set. Further, registration of places or areas may be keyed to specific days or to specific times of the day. An individual may want to register his or her home only for authorized use during the weekends, when that individual knows he or she may need to work from home. The route to and from work may be registered for those times of the day which the user knows he or she is more likely to be commuting. Even further, if the user knows that he or she will be traveling away on business, the user may pre-register the location to which he or she will be traveling, and may register for only those days on which he or she expects to actually be there.

Detection of the individual's actual location when he or she attempts to log-in to a computer network is accomplished by activation and tracking of a locating device which the individual shall have with them. The individual may either be personally carrying the locating device, or it may be attached to, or an integral part of, the computer terminal (whether portable or fixed) from which the individual is logging-in. The locating device itself is a transmitting and receiving device capable of both sending and receiving a location signal. The transmission of the locating signal may, of course, be continuous or intermittent, and may be digital and/or analog in nature.

Activation and initial tracking of the locating device is triggered by the central computer of the network at the time of log-in. Further explanation of the activation and tracking sequence will be given below with reference to the drawings.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 893

Referring to Figure 1, a central computer may have associated with it a network, which from the central computer's perspective is co-located with that computer. In step 100 an individual who will have a need to log-in to the central computer from a site remote from or not co-located with the central computer will pre-register one or more locations from which he or she will want to log-in. Registration of such log-in sites will preferably occur at the location of the central computer using a controller that interfaces with the central computer. Alternatively, such registration may be accomplished from a secure remote site. Once the sites for remote access have been input to the central computer, in step 110 an approval process for each location input will be implemented. As stated above, there may be various approval processes. Preferably someone having a position of authority and/or responsibility for overseeing computer security will give final approval for remote access sites. Further, each site may be designated as "dormant" until an attempt to log-in is made from that remote site. Once a log-in occurs from a site, the site's status may be changed to "active" and notification of the log-in and use of the site may be sent to the appropriate persons (i.e., persons in charge of computer security), and perhaps including the site's registrant. Further, an "active" site which has not been used for a pre-set period of time may be changed back to a "dormant" state. Such classification of sites can be helpful in keeping track of which sites have and/or have not been used and may further help to maintain security.

Once a log-in location has been pre-registered and approved, an individual may access the central computer from that location by logging-in. In step 120 the individual seeking remote access to the central computer and network will log-in in the normally accepted fashion. For instance, the individual will establish contact with the central computer and can present his or her identifying code and password. It is to be understood, of course, that the present invention can be used with any type of log-in procedure, and is not limited to a log-in procedure which uses an identifier and passcode. Further, it should be noted that once a location is registered and approved, as in steps 100 and 110 explained above, the individual need not register that location each time he or she wishes to log-in from that location. On the contrary, the central computer can store the registered and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 894

approved location for future use. In other words, steps 100 and 110 need not be repeated each time the method of the present invention is to be utilized. It may be, however, that re-registration of locations will be required on the basis of some pre-selected criteria, and thus steps 100 and 110 will need to be repeated. For instance, re-registration of a location may be required after a certain period of time has elapsed, after a certain number of log-ins from that location have occurred, after a certain total number of system log-ins have occurred, or any other similar criterion.

In step 130, once the central computer is contacted by an attempted log-in, the central computer will identify on the basis of at least one parameter who the individual attempting to log-in purports to be and will activate the location device associated with that individual. In other words, if the central computer determines that the parameters of the identifier and password submitted in the log-in are associated with a computer user named "Tom", then the computer will activate the location device associated with "Tom" and which "Tom" carries around with him. It is to be understood, of course, that identification of the location device to be activated can be accomplished by any method and on the basis of any parameters which assure that the proper location device will be activated. For instance, parameters used in the log-in and subsequent activation may be on the basis of voice recognition, body heat signature, retinal scan, fingerprint scan, and/or visual observation, etc.

Further, actual activation of the location device can be carried out by any method, as long as the locating device is functionally activated. For instance, activation can be accomplished though radio signals, electrical signals, and/or infrared signals. Preferably the location device will be activated through a medium separate from that which the individual is using to log-in to the central computer. That is, for example, if the individual attempting to log-in is doing so over the Internet, the locating device can be activated through the use of satellite relays.

Upon activation, in step 140, the locating device transmits a location signal. Transmission of the location signal can be by any medium which ensures that the location signal is received by the central computer. For example, the location signal can be transmitted via airwave and relayed by satellite, or

IPR2024-00027
CrowdStrike Exhibit 1002 Page 895

through land-line using the Internet as a relay. The location signal itself can be any type of signal which is capable of carrying the location data and of being transmitted and received. For instance the signal can be radio wave, infrared, or even microwave. Preferably the location signal is broadcast as a radio wave in either a digital or analog format.

In step 150, the broadcast location signal is received by the central computer and a determination of the location of the locating device is made. In order to make the determination, the location signal may act as a homing beacon or may contain location data (coordinates). If the location signal acts as a homing beacon for the location of the location device, the central computer can determine the location of the locating device. If the locating signal contains location data, that is, the actual location (coordinates) of the locating device, then the location device itself can determine its own location. Either way, position detection will need to be accomplished and it is acceptable that any such position detection method or system be utilized. Preferably, the Global Positioning System is used.

Upon a determination of the location of the locating device, in step 160 the central computer decides if the locating device's location is at, or within a pre-determined proximity of, a pre-registered location. If the location is determined to be valid, log-in will be completed. If the location is not valid, log-in will be terminated. This decision step, then, determines whether access will be granted or denied. If the log-in is allowed to be completed, in step 170 the individual logging-in may then access the data files of the central computer. If the log-in is not allowed, in step 180 the connection is terminated and the central computer can generate appropriate messages to the appropriate parties that an unauthorized log-in was attempted.

In the case where the log-in is allowed because the locating device was determined to be at a pre-registered location, periodic updates of the location of the locating device may be accomplished. This ensures that the locating device stays with the individual who has logged-in, and can also act as a way of checking the original determination of the location of the locating device. Further, if any discrepancies occur in the subsequent updates, the central computer can terminate or restrict access. Lastly, an initial log-in from an authorized site can be used to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 896

fine-tune the location's coordinates, if necessary, so that the system can be more accurate.

It should be noted that the central computer may also at any time send a message to the individual identified in step 130 that he or she has been identified as attempting to log-in and/or has been granted access to log-in. Thus if the individual identified in step 130 is at a pre-registered location, but is in fact not logging-in to the central computer, that individual can notify the appropriate personnel and access to the unauthorized individual in fact logging-in can be denied and/or terminated. Messages may be sent in any fashion which will reach the authorized individual identified in step 130. For instance, a message may be sent via telephone, pager, priority e-mail, etc.

Referring to Figure 2, transmission of the location signal is shown. In this example, the central computer 1 communicates with remote computer 2 via communication medium 4, and with location device 3 via communication medium 6. Thus, when an individual attempts to log-in to the central computer 1 using remote computer 2, the central computer 1 sends an activation signal by communication medium 6 to the location device 3. Communication medium 6 uses satellite system 5 for relay of communication. In response, location device 3 sends location signal 7 via communication medium 6 to central computer 1.

It should be noted that other information can be sent along with the location signal. For instance, information which might be sent might include a "time stamp". Such a "time stamp" could be utilized as an assurance that the location signal is being sent from the location indicated by it. The central computer could be synchronized to the GPS atomic clock and determinations of how long the location signal took to transmit could be made. Also, for instance, a passcode for the location device could be sent. A separate passcode for the location device would ensure that the proper location device was transmitting the location signal.

Thus, as can be seen from the foregoing description, an additional layer of computer security can be added to present computer security systems through the use of the present invention. Further, implementation of the present invention would require only nominal system adjustments.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 897

In the foregoing description, the method and apparatus of the present invention have been described with reference to a specific example. It is to be understood and expected that variations in the principles of the method and apparatus herein disclosed may be made by one skilled in the art and it is intended that such modifications, changes, and substitutions are to be included within the scope of the present invention as set forth in the appended claims. The specification and the drawings are accordingly to be regarded in an illustrative rather than in a restrictive sense.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 898

Claims:

1. A method for enhancing computer security using a location device, comprising the steps of:

registering at least one remote log-in location for a computer network;

registering a log-in contact to the computer network;

commanding activation of the location device upon establishment of the log-in contact;

receiving a location signal from the location device;

determining a location of the location device on the basis of the received location signal; and

determining whether the location of the location device is an authorized location with reference to the registered information that identifies the at least one authorized remote log-in location.

2. The method according to claim 1, further comprising the step of:

approving the registration of the at least one remote log-in location.

3. The method according to claim 1, further comprising the step of:

identifying the location device to be activated upon at least one parameter contained in the log-in contact.

4. The method according to claim 1, further comprising the step of:

determining an update of the location of the location device.

5. The method according to claim 1, further comprising the step of:

sending a message to an individual to whom the location device is identified as belonging.

6. The method according to claim 1, wherein activation of the location device is accomplished by radio wave.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 899

7. The method according to claim 1, wherein the transmitted location signal may contain a plurality of data.

8. The method according to claim 7, wherein the plurality of data are location coordinates derived from a Global Positioning System.

9. The method according to claim 1, wherein the transmitted location signal contains a "time stamp".

10. The method according to claim 1, wherein access to the computer network is granted if the location of the location device matches the at least one remote registered log-in location.

11. The method according to claim 1, wherein access to the computer network is denied if the location of the location device does not match the at least one remote registered log-in location.

12. A method for enhancing computer security using a location device, comprising the steps of:

storing information identifying at least one authorized remote log-in location for log-in to a computer;

establishing a log-in contact with the computer;

activating transmission of a locating signal from the location device upon the log-in contact;

determining a location of the location device on the basis of the locating signal; and

determining whether the location of the locating device corresponds to the at least one remote log-in location.

13. The method according to claim 12, further comprising the step of:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 900

securing authorization for the storing of the at least one remote log-in location.

14. The method according to claim 12, further comprising the step of:

identifying the location device from which the location signal is to be activated upon at least one parameter contained in the log-in contact.

15. The method according to claim 12, further comprising the step of:

determining an update of the location of the location device.

16. The method according to claim 12, further comprising the step of:

sending a message to an individual to whom the location device is identified as belonging.

17. The method according to claim 12, wherein activation of the location device is accomplished by radio wave.

18. The method according to claim 12, wherein the transmitted location signal may contain a plurality of data.

19. The method according to claim 18, wherein the plurality of data are location coordinates derived from a Global Positioning System.

20. The method according to claim 12, wherein the transmitted location signal contains a "time stamp".

21. The method according to claim 12, wherein computer access is granted if the location of the location device matches the at least one authorized remote log-in location.

22. The method according to claim 12, wherein computer access is denied if the location of the location device does not match the at least one authorized remote log-in location.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 901

23. A method for enhancing computer security using a location device, comprising the steps of:

receiving a log-in contact;

commanding activation of the location device upon receipt of the log-in contact;

determining a location of the location device on the basis of a received location signal; and

determining whether the location of the location device corresponds to an authorized remote log-in location.

24. The method according to claim 23, further comprising the step of:

storing information identifying at least one authorized remote log-in location for log-in to a computer.

25. The method according to claim 24, further comprising the step of:

securing authorization for the storing of the at least one remote log-in location.

26. The method according to claim 23, further comprising the step of:

identifying the location device to be activated upon at least one parameter contained in the log-in contact.

27. The method according to claim 23, further comprising the step of:

determining an update of the location of the location device.

28. The method according to claim 23, further comprising the step of:

sending a message to an individual to whom the location device is identified as belonging.

29. The method according to claim 23, wherein activation of the location device is accomplished by radio wave.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 902