
Network Intrusion Detection

Intrusion detection is a new, retrofit approach for providing a sense of security in existing computers and data networks, while allowing them to operate in their current "open" mode.

■■■■■■■■■■

Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt

BISWANATH MUKHERJEE is an associate professor of computer science at the University of California, Davis.

L. TODD HEBERLEIN is a postgraduate researcher in the Computer Science Department at UC Davis.

KARL N. LEVITT is a professor of computer science at UC Davis.

¹Authentication is the process of determining whether or not an activity is a genuine one. It is a very desirable security service and is an important property of a secure network or computer system. Data and communications integrity can be directly built on authentication mechanisms. Identification is the process of determining whether someone is truly the person who he says he is.

Intrusion detection is a new, retrofit approach for providing a sense of security in existing computers and data networks, while allowing them to operate in their current "open" mode. The goal of intrusion detection is to identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators. The intrusion detection problem is becoming a challenging task due to the proliferation of heterogeneous computer networks since the increased connectivity of computer systems gives greater access to outsiders and makes it easier for intruders to avoid identification. Intrusion detection systems (IDSs) are based on the beliefs that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Typically, IDSs employ statistical anomaly and rule-based misuse models in order to detect intrusions.

A number of prototype IDSs have been developed at several institutions, and some of them have also been deployed on an experimental basis in operational systems. In this paper, several host-based and network-based IDSs are surveyed, and the characteristics of the corresponding systems are identified. The host-based systems employ the host operating system's audit trails as the main source of input to detect intrusive activity, while most of the network-based IDSs build their detection mechanism on monitored network traffic, and some employ host audit trails as well. An outline of a statistical anomaly detection algorithm employed in a typical IDS is also included.

Introduction

A secure computer or network system should provide the following services — *data confidentiality*, *data and communications integrity*, and assurance against *denial-of-service* [1, 2]. Data confidentiality service protects data against unauthorized disclosure. Release of a message's content to unauthorized users is a compromise which this service should protect. Data and communications integrity service is concerned with the accuracy, faithfulness, non-corruptibility, and believability of

information transfer between peer entities (including computers connected by a network). This service must ensure correct operation of the system hardware and firmware, and it should protect against unauthorized modification of data and labels. Denial-of-service is a threat, and assurance against denial-of-service is an important security service [3]. A denial-of-service condition is said to exist whenever the system throughput falls below a pre-established threshold, or when access to a (remote) entity is unavailable. While such attacks are not completely preventable, it is often desirable to reduce the probability of such attacks below some threshold.

The conventional approach to secure a computer or network system is to build a "protective shield" around it. Outsiders who need to enter the system must identify and authenticate themselves — commonly known as the Identification & Authentication (I&A) problem.¹ Also, the shield should prevent leakage of information from the protected domain to the outside world. Mandatory access control techniques (e.g., cryptography-based) might be used in the design of such secure systems [1].

There are a number of limitations to this prevention-based approach for computer and network security, as outlined below.

- It is difficult, perhaps impossible, to build a useful system which is absolutely secure. That is, the possible existence of some design flaw in a system with a large number of components cannot be excluded. In addition, one cannot rule out the occurrence of administrative flaws such as misconfiguration of equipment when bought from vendor, errors due to backward compatibility of vendor equipment, and poor administrative policies and practices.
- It is impractical to assume that the vast existing infrastructure of (possibly insecure) computer and network systems will be scrapped in favor of new, secure systems since a tremendous investment into our current infrastructure has already been made.
- The prevention-based security philosophy constrains a user's activities; the current "open" mode

of operation of most systems is regarded by many to be a highly-useful environment for promoting user productivity.

- Crypto-based systems cannot defend against lost or stolen keys, and against cracked passwords.
- Finally, a secure system can still be vulnerable to insiders misusing their privileges since it cannot fully guard against the insider threat, i.e., users who abuse their privileges. (Systems with mandatory access controls, however, can reduce the risks of some kinds of insider attacks.)

Around the mid-'80s, an alternative approach, called intrusion detection, for providing a different notion of security in computer systems was proposed [4]. The basic arguments in favor of this concept are those outlined in the previous paragraph, namely, that not only abandoning the existing and huge infrastructure of possibly-insecure computer and network systems is impossible, but also replacing them by totally-secure systems may not be feasible or cost effective. That is, our computers and networks may be under attack; but an intrusion detection system based on a retrofit technology should be able to detect such attacks, preferably in real time (i.e., when the attacks are in progress). Typically, an intrusion detection systems (IDS) alerts a system security officer (SSO) when it detects an attack. This approach is gaining increasing momentum and acceptance, and a number of prototype IDSs — some for a single host and others for several hosts connected by a network — have been built at several institutions.

Intrusion detection is defined to be the problem of identifying individuals² who are using a computer system without authorization (i.e., “crackers”) and those who have legitimate access to the system but are abusing their privileges (i.e., the “insider threat”). Generally, an intrusion would cause loss of confidentiality, loss of integrity, denial of resources, or unauthorized use of resources. Some specific examples of intrusions that concern system administrators include:

- Unauthorized modifications of system files so as to permit unauthorized access to either system or user information.
- Unauthorized access or modifications of user files/information.
- Unauthorized modifications of tables or other system information in network components (e.g., modifications of router tables in an internet to deny use of the network).
- Unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or perhaps through the unauthorized use of existing accounts).

An example intrusion scenario is included at the end of this section.

Detecting attacks requires the use of a model of intrusion, namely, what should the IDS look for? Currently, two types of models are employed in IDSs. The first model hypothesizes its detection upon the profile of a user's (or a group of users') normal behavior. It statistically analyzes parameters of the user's current session, compares them to the profile representing the user's normal behavior, and reports “significant” deviations to a SSO. Here, significant is defined as a threshold set by the specific model or by the SSO. A typical IDS may report the “Top Ten” most suspicious

sessions to the SSO [5]. Because it catches sessions which are not normal, it is referred to as an “anomaly” detection model.

The second type of model bases its detection upon a comparison of parameters of the user's session and the user's commands to a rule-base of techniques used by attackers to penetrate a system. Attack signatures (i.e., known attack methods) are what this model looks for in the user's behavior. Since this model looks for patterns known to cause security problems, it is called a “misuse” detection model.

A number of IDSs base their design on analyzing the host operating system (OS)'s audit trails. Their examples include AT&T's ComputerWatch [6], TRW's Discovery [7, 8], Haystack Laboratory's HAYSTACK system [5], SRI International's Intrusion Detection Expert System (IDES) [9, 10, 11], Planning Research Corporation's Information Security Officer's Assistant (ISOA) [12, 13], National Security Agency's Multics Intrusion Detection and Alerting System (MIDAS) [14], and Los Alamos National Laboratory's Wisdom & Sense (W&S) [15] and Network Anomaly Detection and Intrusion Reporter (NADIR) [16]. Some of the basic algorithms employed in these systems include evaluation of a weighted multinomial function to detect deviations from normal behavior, a covariance-matrix-based approach for profiling normal behavior, and rule-based expert system approach to detect violations of security policy.

Early IDS models were designed to monitor a single host. However, more recent models accommodate the monitoring of a number of hosts interconnected by a network, e.g., ISOA, IDES, and UC Davis' Network Security Monitor (NSM) and Distributed Intrusion Detection System (DIDS). Some of these systems (ISOA and IDES) transfer the monitored information (host audit trails) from the monitored hosts to a central site for processing. Others (NSM, DIDS) monitor the network traffic flow as well, as part of their intrusion detection algorithms.

An Example Intrusion Scenario

A description of a real attack that occurred several months ago and was detected by our Network Security Monitor (NSM) [17] provides a good example of the types of attacks that occur regularly and must be detected. Pertinent facts include the following:

- At least ten different computers were involved.
- The computers were managed by eight sets of system administrators distributed over seven different sites, three states, and two countries.
- The attack exploited a number of different vulnerabilities in a number of different computer systems.

The attack took place in several stages over several days.

1) The initial phase of the attack included: a series of “doorknob-rattling” operations (namely, the use of common account_name/password combinations to break-in) from COMPANY1.com, resulting in a successful break-in into shark.SCHOOL2.edu by exploiting a flaw; importing a Trojan login program from omen.SCHOOL3.edu and installing it in shark; and followed (on the next day) by a login from revir.SCHOOL1.edu into shark.SCHOOL2.edu by the Trojan login program installed the previ-

■ ■ ■ ■ ■
Detecting
attacks
requires
the use
of a model
of intrusion:
what should
the IDS
look for?

² Typically, these individuals are users, but they may be hosts or programs (in case of machine attacks) as well.

■ ■ ■ ■ ■

The Computer-Watch audit trail analysis tool provides a significant amount of audit data reduction and limited intrusion-detection capability.

ous day. Apparently this attempt is merely testing if the Trojan still existed, and the intruder quickly logs off.

2) The second element of the attack observed is another login to *shark* exploiting the Trojan login program; however, this login comes from *bear.SCHOOL4.edu*. Although this attack came from a different place, we are confident that this is the same person. This is based on two facts: the Trojan horse was installed only the night before; and the special password used was specific to *shark*, i.e., although other Trojan horses have been discovered, the password selected and set the night before is unique to *shark*. The intruder then uses *shark* as a platform from which to attack other computer systems.

3) The intruder exploits a hole in a *.rhosts* file on a computer at a well-known school on the east coast, *next.SCHOOL6.edu*, and logs in as *uucp*. Once on *next*, the intruder executes a program granting him root privileges.

4) As root, the intruder is able to exploit the fact that another computer's file system, *kropotkin.SCHOOL7.edu*, is mountable by *next*. Once the intruder is able to mount *kropotkin*, he is able to examine and manipulate the file system without having to login to *kropotkin*. The intruder installs another hole into *kropotkin* that allows anyone to login to the account *tami* from anywhere.

5) As it turns out, the home directory for user *tami* on *kropotkin* is the same on two other hosts at *SCHOOL7*, *wombat.SCHOOL7.edu* and *SCHOOL7.edu*. This fact gives the intruder access to these machines as well. After moving about the different *SCHOOL7* computer systems, the intruder returns to *shark* at *SCHOOL2*.

6) The intruder next attacks a computer at *SCHOOL5*, called *SCHOOL5.edu*, by exploiting a Trojan login program previously installed.

7) Using *SCHOOL5* as a platform, the intruder attacks a computer in Canada, *polyv.COUNTRY2*, by exploiting a Trojan login program there as well. The intruder notices the system administrator currently active, and he exits *polyv*.

8) After extensively examining *SCHOOL5*'s file system, the intruder returns again to *shark* at *SCHOOL2*.

9) From *shark*, the intruder breaks into a computer at *COMPANY2*, previous *COMPANY2.com*, by exploiting a "+" in the *.rhosts* file for the account *me*. The intruder, apparently satisfied that this hole is still intact, returns back to *shark*.

10) The intruder again breaks into *polyv.COUNTRY2*, and once again, his visit is short.

11) Finally, after more than six hours of attacking various computer systems, the intruder exits *shark* to return to *bear.SCHOOL4.edu*.

Host-Based Intrusion Detection Systems

An early abstract model of a typical IDS was proposed in [4]. Since then, a number of IDSs have been designed and deployed. A large number of IDSs employ their host OS's audit trail as the main source of input for detecting intrusions. Such systems are surveyed in this sec-

tion. For each IDS surveyed, we provide an overview of the system, an outline of the system's organization, and a discussion on how the system operates.

ComputerWatch

Overview — The ComputerWatch audit trail analysis tool provides a significant amount of audit data reduction and limited intrusion-detection capability [6]. The amount of data viewed by an SSO is reduced while minimizing the loss of any informational content. Data reduction is performed by providing a mechanism for examining different views of the audit data based on information relationships.

ComputerWatch, designed for the System V/MLS operating system, was written to assist, but not replace, the SSO. The tool uses an expert system approach to summarize security sensitive events and to apply rules to detect anomalous behavior. It also provides a method for detailed analysis of user actions in order to track suspicious behavior.

System Organization — Audit trail records can be analyzed either by the SSO interactively, or in batch mode for later review, i.e., ComputerWatch does no real-time analysis of events. There are three levels of detection statistics, namely, system, group, and user. Statistical information for system-wide events is provided in a summary report. Statistical information for user-based events is provided by detection queries. Statistical information for group-based events will be a later enhancement [6].

System Operation — ComputerWatch provides a System Activity Summary Report for the SSO. This report contains summary information describing the security-relevant activities occurring on the system. The report can indicate what types of events need closer examination. The SSO can also perform his own analysis on the data. Expert system rules are used to detect anomalies or simple security breaches. The rules are fired when an equation is satisfied and when the rules in its predecessor list have been fired as well.

The detection queries that are provided have been designed to assist the SSO in detecting "simple" system security breaches. These security breaches may involve intrusion, disclosure, or integrity subversion. The detection queries display similar security-relevant system activities as those that are described in the summary report, but at a user level. A SQL-based query language is provided to allow the SSO the capability to design custom queries for intrusion detection.

Discovery

Overview — Discovery is an expert system tool developed by TRW for detecting unauthorized accesses to its credit database [7, 8]. The Discovery system itself is written in COBOL, while the expert system is written in an AI shell. Both run on IBM 3090s. Their goal is not to detect attacks on the operating system, but to detect abuses of the application, namely, the credit database.

System Organization — TRW runs a database that contains the credit histories of 133 million consumers. It is accessed more than 400,000 times a day using 150,000 different access codes,

many of which are used by more than one person [7, 8], but these numbers are expected to have increased by now. The database is accessed in three ways: on-line access by TRW customers who query consumers' credit information, monthly updates from accounts receivable data received on magnetic tape, and modifications to correct errors and inaccuracies. The Discovery system examines each of these processes for unauthorized activity.

Discovery is a statistical inference system which looks for patterns in the input data. Its targets include hackers, private investigators, and criminals. It is designed to detect three types of undesired activity, namely, accesses by unauthorized users, unauthorized activities by authorized users, and invalid transactions. Processing of the audit data is performed in daily batches.

System Operation — 1) Customer Inquiries. Discovery's processing sequence is as follows. First, records with invalid formats are discarded. Valid records are then sorted and processed by a pattern recognition module. Inquiries are compared to both the standard inquiry profile and a model of illegitimate access. Access codes which are suspected of having been misused can also be flagged for tighter scrutiny.

The system produces a user profile for each customer by type-of-service and access method. These profiles are updated daily. The system generates statistical patterns based on the variables in each inquiry (e.g., presence or absence of a middle initial), access characteristics (e.g., time of day), and characteristics of a credit record (e.g., geographic area).

Each variable has a tolerance, established by accumulated patterns, within which the daily activity should fall. Three types of comparison are made: each inquiry with the global pattern, each subscriber's daily pattern with the global pattern, and a subscriber's pattern with an industry pattern. The system's output is an exception data file that lists the reasons for the exceptions, as well as a report module. Investigative data is also stored in a database and may be retrieved using a query language.

Initial production runs of the expert system produced large numbers of exceptions. Some of these were traced to variations due to time of day, etc. Heuristics based on analysis of actual cases are also being included in the expert system.

2) Database Update. Several factors in the incoming data from customers are measured by a COBOL program. Data is entered into the database only if statistical comparisons with previously reported data are within a pre-defined tolerance. Data that is rejected by the statistical analysis is submitted to an expert system for further validation, and is entered into the database if passed by the expert system.

3) Database Maintenance. The credit database may be modified by TRW operators to correct errors and inaccuracies. An expert system designed to monitor the maintenance process performs statistical analysis of maintenance transactions and analyzes each credit record's maintenance history.

Discovery has detected and isolated unauthorized accesses to the database, masqueraders, and invalid inquiries. It has also provided investigators with concise leads on illegitimate activity.

Several of the deviations that have been discovered were found to be caused by customers changing their access methods and systems. The expert system allows TRW to apply a consistent security policy in the update process. A beneficial side-effect of the system is the compilation of purchasing patterns for each customer, which is useful for marketing purposes.

HAYSTACK

Overview — HAYSTACK was initially designed to be a system for helping Air Force Security Officers detect misuse of Unisys 1100/2200 mainframes used at Air Force Bases for routine "unclassified but sensitive" data processing [5]. HAYSTACK software reduces voluminous system audit trails to short summaries of user behaviors, anomalous events, and security incidents. This reduction enables detection and investigation of intrusions, particularly by insiders (authorized users).

In addition to providing audit trail data reduction, HAYSTACK attempts to detect several types of intrusions: attempted break-ins, masquerade attacks, penetration of the security system, leakage of information, denial of service, and malicious use. HAYSTACK's operation is based on behavioral constraints imposed by official security policies and on models of typical behavior for user groups and individual users.

System Organization — The initial HAYSTACK system consisted of two program clusters, one executing on the Unisys 1100/2200 mainframe, and the other executing on a 386-based PC running MS-DOS and the ORACLE database management system [5]. Data is transferred from the mainframe to the PC by magnetic tape or electronic file transfer over a communications line. Performance-wise, it has been found that a typical day's worth of audit data can be processed within a few hours on the PC.

The preprocessor portion of HAYSTACK (which runs on the mainframe) is a straightforward COBOL application that selects appropriate audit trail records from the Unisys proprietary audit trail file as input, extracts the required information, and reformats it into a standardized format for processing on the PC. Software on the PC is written in C, embedded SQL, and ORACLE tools. It processes and analyzes the audit trail files, helps the SSO maintain the databases that underlie HAYSTACK, and gives the SSO additional support for his investigations.

System Operation — HAYSTACK helps an SSO detect intrusions (or misuse) in three different ways.

1) Notable Events. HAYSTACK highlights notable single events for review. Events that modify the security state of the system are reported, along with explanatory messages. This includes both "successful" and "unsuccessful" events that affect access controls, user-ids, and group-ids.

2) Special Monitoring. The SSO may "tag" particular security "subjects" and "objects" for special monitoring. This is analogous to setting an alarm to go off when a particular user-id is active, or when a particular file or program is accessed. This alarm may also increase the amount of reporting of the user's activity.

■ ■ ■ ■ ■
HAYSTACK
software
reduces
voluminous
system audit
trails to
short
summaries
of user
behaviors,
anomalous
events, and
security
incidents.

■ ■ ■ ■ ■

The overall goal of IDES is to provide a system-independent mechanism for the real-time detection of security violations.

3) Statistical Analysis. HAYSTACK performs two different kinds of statistical analysis. The first kind of statistical analysis yields a set of "suspicion quotients." These are measures of the degree to which the user's aggregate session behavior resembles one of the target intrusions which HAYSTACK is trying to detect.

About two dozen "features" (behavioral measures) of the user's session are monitored on the Unisys system, including time of work, number of files created, number of pages printed, etc. Given a list of the session features whose values were outside the expected ranges for the user's security group, plus the estimated significance of each feature violation for detecting a target intrusion, HAYSTACK computes a weighted multinomial "suspicion quotient" which signifies how closely that session resembles a target intrusion for the user's security group. The suspicion quotient is therefore a measure of the "anomalousness" of the session with respect to a particular weighting of features. HAYSTACK emphasizes that such suspicions are not "smoking guns," but are rather hints or hunches to the SSO that the session may warrant further investigation. Such a statistical anomaly detection algorithm is treated in greater detail in the section on an intrusion detection algorithm case study later in this article.

The second kind of statistical analysis detects variation within a user's behavior by looking for significant changes ("trends") in recent sessions compared to previous sessions.

Intrusion-Detection Expert System (IDES)

Overview — The Intrusion-Detection Expert System (IDES) developed at SRI International is a comprehensive system that uses complex statistical methods to detect atypical behavior, as well as an expert system that encodes known intrusion scenarios, known system vulnerabilities, and the site-specific security policy [9-11].

The overall goal of IDES is to provide a system-independent mechanism for the real-time detection of security violations. These violations can be initiated by outsiders who attempt to break into a system or by insiders who attempt to misuse their privileges. IDES runs independently on its own system (currently a Sun workstation) and processes the audit data received from the system being monitored.

System Organization — The IDES prototype uses a subject's historical profile of activity to determine whether its current behavior is normal with respect to past or acceptable behavior. Subjects are defined as users, remote hosts, or target systems. A profile is a description of a subject's normal (i.e., expected) behavior with respect to a set of intrusion-detection measures. IDES monitors target system activity as it is recorded in audit records generated by the target system. Due to the fact that these profiles are updated daily, IDES is able to adaptively learn a subject's behavior patterns; as users alter their behavior, the profiles change to reflect the most recent activity. Rather than storing the tremendous amount of audit data, the subject profiles keep only certain statistics such as frequency tables, means, and covariances.

IDES also includes an expert-system component that is able to describe suspicious behavior that is

independent of whether a user is deviating from past behavior patterns. The expert system contains rules that describe suspicious behavior based on knowledge of past intrusions, known system vulnerabilities, or the site-specific security policy.

The IDES comprehensive system is considered to be loosely coupled in the sense that the decisions made by the two components are independent. While the two components share the same source of audit records and produce similar reports, their internal processing is done separately. The desired effect of combining these two separate components is a complementary system in which each approach will help to cover the limitations of the other.

System Operation — The system has two major components as discussed below.

1) The Statistical Anomaly Detector (IDES/STAT) [11]. In order to determine whether or not current activity is atypical, IDES/STAT uses a deductive process based on statistics. The process is controlled by dynamically-adjustable parameters that are specific to each subject. Audited activity is described by a vector of intrusion-detection variables that correspond to the measures recorded in the profiles. As each audit record arrives, the relevant profiles are retrieved from the knowledge base and compared with the vector of intrusion-detection variables. If the point defined by the vector of intrusion-detection variables is sufficiently far from the point defined by the expected values, with respect to the historical covariances for the variables stored in the profiles, then the record is considered anomalous. The covariance-matrix-based approach, however, has turned out to be compute-intensive, and recent versions of IDES have dropped the covariance-based computations [18].

The procedures are not only concerned with whether an audit variable is out of range, but also with whether an audit variable is out of range relative to the values of the other audit variables. IDES/STAT evaluates the total usage pattern, not just how the subject behaves with respect to each measure considered singly.

2) The Expert System. The IDES expert system will make attack decisions based on information contained in the rule-base regarding known attack scenarios, known system vulnerabilities, site-specific security information, and expected system behavior. It will, however, be vulnerable to intrusion scenarios that are not described in the knowledge base.

The expert system component is a rule-based, forward-chaining system. A production-based expert system tool (PBEST) has been used to produce a working system. The PBEST translator is used to translate the rule-base into C language code, which actually improves the performance of the system over using an interpreter. As the size of the rule-base increases, the processing time will also increase since the functions that implement the rules must search longer lists.

Information Security Officer's Assistant (ISOA)

Overview — The Information Security Officer's Assistant (ISOA) is a real-time security monitor implemented on a UNIX-based workstation that supports automated as well as interactive audit trail analysis [12, 13]. This monitor provides a sys-

tem for the timely correlation and merging of disjoint details into an assessment of the current security status of users and hosts on a network. The audit records, which are indicators of actual events, are correlated with known indicators (i.e., expected events) organized in hierarchies of concern, or security status.

ISOA's analysis capabilities include both statistical as well as expert system components. These components cooperate in the automated examination of various "concern levels" of data analysis. As recognized indicators (sets of indicators) are matched, concern levels increase and the system begins to analyze increasingly detailed classes of audit events for the user or host in question.

System Organization — The monitoring of events that do not constitute direct violation of the security policy requires a means to specify expected behavior on a user and host basis. The expected behavior can be represented in profiles that specify thresholds as well as associated reliability factors for discrete events. The observed events can then be compared to expected measures, and deviations can be identified by statistical checks of expected versus actual behavior. ISOA profiles also include a historical abstract of monitored behavior (e.g., a record of how often each threshold was violated in the past), and inferences that the expert system has made about the user. Hosts as well as individual users are monitored.

Events that cannot be monitored by examining thresholds make it necessary to effect a higher-order analysis that is geared towards correlating and resolving the meaning of diverse events. The expert system analysis component can specify the possible relationships and implied meaning of diverse events using its rule-base. Where statistical measures can quantify behavior, the rule-based analysis component can answer conditional questions based on sets of events.

System Operation — The underlying processing model of ISOA consists of a hierarchy of concern levels constructed from indicators. Analysis is structured around these indicators to build a global view of the security status for each monitored user and host. The indicators allow modeling and identification of various classes of suspicious behavior, such as aggregator, imposter, misfeasor, etc.

Two major classes of measures are defined: real-time and session. The real-time measures require immediate analysis, while session measures require (at minimum) start-of-session and end-of-session analysis.

ISOA supports two classes of anomaly detection: preliminary and secondary. Preliminary anomaly detection takes place during the collection of the audit data (i.e., in real time). Predetermined events trigger an investigation of the current indicator or event of interest. If further analysis is warranted, the current parameters are checked against the profiles for real-time violations or deviations from expected behavior.

Secondary anomaly detection is invoked at the end of a user login session or when required for resolution. The current session statistics are checked against the profiles, and session exceptions are determined.

When the expert system is notified that the state of indicators has changed significantly, it attempts to resolve the meaning of the current state of indicators. This is done by evaluating the appropriate

subset of the overall rule-base, which consists of a number of individual rules that relate various indicator states with one another and with established threat profiles. The end result of anomaly resolution is presented to the SSO in the form of a graphical alert, an advice, and an explanation as to why the current security level is appropriate.

Multics Intrusion Detection and Alerting System (MIDAS)

Overview — The Multics Intrusion Detection and Alerting System (MIDAS) is an expert system which provides real-time intrusion and misuse detection for the National Computer Security Center's networked mainframe, Dockmaster, a Honeywell DPS-8/70 Multics computer system [14].

MIDAS has been developed to employ the basic concept that statistical analysis of computer system activities can be used to characterize normal system and user behavior. User or system activity that deviates beyond certain bounds should then be detectable.

System Organization — MIDAS consists of several distinct parts. Those implemented on Dockmaster itself include the command monitor, a preprocessor, and a network-interface daemon. Those that are installed on a separate Symbolics Lisp machine include a statistical database, a MIDAS knowledge base, and the user interface.

The command monitor captures command execution data that is not audited by the Multics system, the preprocessor transforms Dockmaster audit log entries into a canonical format, and the network-interface daemon controls communications. The statistical database records user and system statistics, the knowledge base consists of a representation of the current fact base and rule-base, and the user interface provides communication between MIDAS and the SSO.

An expert system utilizes a forward-chaining algorithm with four tiers (generations) of rules. The firing of some combination of rules in one tier can cause the firing of a rule in the next tier. The higher the tier, the more specific the rules become in regards to the possibility of attacks.

MIDAS keeps user and system-wide statistical profiles that record the aggregation of monitored system activity. The user's (system's) current session profile is compared to the historical profile to determine whether or not the current activity is outside two standard deviations.

System Operation — The logical structure of MIDAS revolves around the rules (heuristics) contained in the rule-base. There are currently three different types of rules which MIDAS employs to review audit data.

1) Immediate Attack. These rules examine a small number of data items without using any kind of statistical information. They are intended to find only those auditable events that are, by themselves, abnormal enough to raise suspicion.

2) User Anomaly. These rules use statistical profiles to detect when a user's behavior profile deviates from previously-observed behavior patterns. User profiles are updated at the end of a user's session if the behavior has changed significantly, and are maintained for each user throughout the life of the account.

■ ■ ■ ■ ■

MIDAS

has been developed to employ the basic concept that statistical analysis of computer system activities can be used to characterize normal system and user behavior.

■ ■ ■ ■ ■

Wisdom and Sense is an anomaly detection system that operates on a Unix (IBM RT/PC) platform and analyzes audit trails from VAX/VMS hosts.

3) System State. These rules are similar to the user anomaly rules, but depict what is normal for the entire system, rather than for single users.

Wisdom and Sense

Overview — Wisdom and Sense (W&S) is an anomaly detection system developed at the Los Alamos National Laboratory [15]. It operates on a UNIX (IBM RT/PC) platform and analyzes audit trails from VAX/VMS hosts. It is an anomaly detection system which seeks to identify system usage patterns which are different from historical norms. It can process audit trail records in real time, although it is hampered by the fact that the operating system may delay writing the audit records.

The objectives of W&S are to detect intrusions, malicious or erroneous behavior by users, Trojan horses, and viruses. The system is based on the presumption that such behavior is anomalous and could be detected by comparing audit data produced by them with that of routine operation.

System Organization — W&S is a statistical, rule-based system. One of its major features is that it derives its own rule-base from audit data. It receives historical audit data from the operating system and processes it into rules. These rules are formed into a forest (i.e., a set of trees). The rules are human-readable, and thus the rule-base may be supplemented or modified by a human expert to correct deficiencies and inconsistencies. The rules define patterns of normal behavior in the system. A W&S rule-base may contain between 10^4 and 10^6 rules, which take 6 to 8 bytes each, and can be searched in about 50 ms. A typical generation of the rule-base takes less than an hour on an inexpensive workstation.

W&S views the universe as a collection of events, each represented by an audit record. Audit log records contain data about the execution of individual processes. Each record consists of a number of fields which contain information such as the invoker (user), the name of the process, its privileges, and system resources utilized.

Data is viewed primarily as categorical, i.e., any field in a record can take one of a number of values. Categorical data is represented as character strings. Continuous data, such as CPU time, is mapped into a set of closed ranges, and then treated as categorical data.

System Operation — 1) Rules. Rules consist of a left-hand side (LHS), which specifies the conditions under which the rule applies; and a right-hand side (RHS) (also referred to as the rule's restriction), which defines what is considered normal under these conditions. The absence of a rule means that everything is considered normal.

The LHS could consist of field values or value ranges, values computed from a series of records (e.g., mean time between events), or subroutines returning a Boolean value. A given rule fires only if an audit record has fields whose values match the LHS and if any subroutines in the LHS return true.

The RHS may take the form of a list of acceptable categorical values for a record field, a list of acceptable ranges of a continuous field, and a list of user-defined functions. Each rule has a grade, which is a measure of its accuracy. Rules which

are more specific, or which represent frequently-occurring patterns with less variability, are given better (i.e., higher) grades.

2) Constructing the Rule-Base. The historical data is first condensed, and then processed through the rule-base generator, which builds the forest of rule trees. At each level, the rule-base consists of nodes designating fields, and nodes designating acceptable values of each field. The rules are generated by repeatedly sorting the data and examining the frequency of field values. The tree is pruned as it is being built by using a number of pruning rules to limit its size.

3) Audit Data Analysis. The "Sense" part of W&S analyzes an activity file using the rule forest. It looks at a record, finds the applicable rules, and computes a figure of merit (FOM) for each field and each transaction. A transaction's FOM is the normalized sum of the grades of failed rules.

A number of transactions may be grouped to form a *thread*. Each thread belongs to a *thread class* that is defined by values of specific audit record fields. Some of the thread classes that are used include: each user-terminal combination, each program-user combination, and each privilege level. A set of operations may be defined for each thread class and carried out whenever a record in the class is processed. A FOM is computed for each thread as a time-decayed sum of the FOM's of its transactions. A transaction, or a thread, is considered anomalous if its FOM is above a pre-defined threshold.

The Sense module also provides an interactive interface to the configuration settings, rule-base maintenance routines, and analysis tools. W&S offers several aids to the task of explaining the meaning and cause of anomalous events. It has undergone operational testing and has detected interesting anomalies even in data originally thought to be free of such events.

Other Related Work

Additional related work can be found in the literature. Some are worth mentioning even though they may not fit in cleanly with our definition of an intrusion detection system. Recall that an IDS performs passive monitoring of computing resource usage, without changing the system's services *per se*.

The AT&T Dragons Approach — The AT&T Bell Labs work [19, 20] deviates from the above definition of an IDS because it replaces standard servers by a variety of trap programs that look for attacks. However, this approach is relevant because it can detect intruders; study the attackers' strategies, tools, and techniques; and alert the SSO accordingly. Specifically, these "proxy servers" are implemented on AT&T Bell Labs' Internet security gateway research.att.com. Except for some servers such as mail, FTP, and telnet, other services are replaced by "dummy servers." (This is partly justified by the widespread existence of security problems in current Internet software [21].) Some of these dummies are "packet suckers" while others are quite specialized. All such servers log the incoming request, attempt to trace it back (namely, employ counter-intelligence approaches to learn more about the source of the attack, e.g., via reverse fingers), and try to distinguish between legitimate users and outside attackers. These

tools have detected a variety of attacks from simple doorknob-rattling (such as guest login) to the more determined (e.g., forged NFS packets). Finally, an interesting chronicle on how an attacker is lured into the machine and how his actions are studied can be found in [20].

Signature Analysis — Some generic approaches for representing and detecting “attack signatures” have been reported [22-24]. One of these methods [22] employs sequential rules that characterize a user’s behavior over time. A rulebase stores patterns of user activity, e.g., a rule can characterize the sequential relationship between security-relevant audit records. The rules can be static (based on security policy) or dynamic (based on time-based inductive learning techniques). Anomalies are detected whenever a user’s activity deviates significantly from those specified in the rules. The main strength of this approach is that it allows adjacent security events to be correlated.

Clustering Techniques — Many of the IDSs discussed above rely on features of system and user behavior as inputs to their analysis algorithms which then determine the likelihood of an intrusion. The choice of these features is quite arbitrary and is based solely on the experience of an expert. A very relevant problem, called “clustering,” is to determine important features to be used in an effective IDS design. This approach could be based upon an investigation of the experimentally-derived effectiveness of the features at classifying users as attackers and non-attackers [25, 26].

Network-Based Intrusion Detection Systems

ISOA and IDES

Early IDS models were designed to support a single host. However, more recent models accommodate the monitoring of a number of hosts interconnected by a network, e.g., ISOA and IDES. These systems (ISOA and IDES) transfer the monitored information (host audit trails) from multiple monitored hosts to a central site for processing. They employ the same algorithms as in the host-based systems. They do not monitor any network traffic.

Network Anomaly Detection and Intrusion Reporter (NADIR)

Overview — Network Anomaly Detection and Intrusion Reporter (NADIR) is a misuse detection system designed for Los Alamos National Laboratory (LANL)’s Integrated Computing Network (ICN) [16]. It is an automated expert system, which streamlines and supplements the manual audit record review performed by the SSO. NADIR compares weekly network activity of individual users and the ICN as a whole, against expert rules that define security policy and improper or suspicious behavior. It reports suspicious behavior to the SSO, and provides tools to allow the SSO to perform followup investigations.

System Organization — The ICN is LANL’s main computer network. It serves nearly 9,000 users and includes computing equipment from super-

computers to terminals, each of which connect to an ICN port. An ICN port belongs to one of four partitions, each defined to operate at a certain security level. That is, a computer can access other computers in its partition or in partitions in lower (less secure) levels. The partitions are linked via a system of dedicated service nodes, namely, Network Security Controller (NSC) that provides user authentication and access control on ICN; Common File System (CFS) that stores data from each partition separately and guards against users in lower-partition machines accessing files stored in higher-partition machines; and Security Assurance Machine (SAM) that authenticates and records all attempts to down-partition files within CFS.

NSC, CFS, and SAM send raw audit records in “home-grown” format to NADIR, which is run on a SUN SPARCstation II. NADIR is implemented using the Sybase relational database management system.

System Operation — NADIR receives raw audit records from NSC, CFS, and SAM, and it generates weekly summaries of both individual user activity and aggregate ICN activity. (An example raw audit record from NSC would contain the partition and ICN number of the machine from which the authentication attempt is generated, plus the partition, classification level, and network component that the user wishes to access.) NADIR has a set of built-in expert rules for misuse detection; these rules are developed through audit analysis and consultation with security experts. NADIR compares weekly summaries with these rules, and assigns a “level-of-interest” to each rule that is triggered. A user’s suspicion level is the sum of the level-of-interest of all rules it triggers. NADIR graphically shows its weekly reports on network usage, and it also highlights the most suspicious users. It can also provide more detailed reports on raw or profiled audit data to assist the SSO.

Network Security Monitor (NSM)

Overview (Advantages of Monitoring Network Traffic) — The Network Security Monitor (NSM) has been developed at the University of California, Davis. The NSM is different from the IDSs discussed earlier in that it does not analyze audit trails [17, 27-29]. The NSM analyzes traffic on a broadcast LAN to detect intrusive behavior. The reasons for this departure from the standard intrusion detection methods are outlined below.

First, although most IDSs are designed with the goal of supporting a number of different operating system platforms, all present audit-trail-based IDSs have only been used on a single operating system at any one time. These systems are usually designed to transform an audit log into a proprietary format used by the IDS [5, 9, 14]. In theory, audit logs from different operating systems need only to be transformed into this proprietary form for the IDS to perform its analysis. An IDS that can simultaneously support multiple operating systems is desirable. On the other hand, standard network protocols exist, e.g., TCP/IP and UDP/IP, which most major operating systems support and use. By using these network standards, the NSM can monitor a heterogeneous set of hosts and operating systems simultaneously.

■ ■ ■ ■ ■
Network
Anomaly
Detection
and Intrusion
Reporter
is an
automated
expert
system that
streamlines
and
supplements
the manual
audit record
review per-
formed by
the SSO.

Component	Description
Connection_ID	Unique integer used to reference this particular connection.
Initiator_address	The internet address of the host which initiated the connection.
Receiver_address	The internet address of the host to which the connection was made.
Service	An integer used to identify the particular service (i.e., telnet or mail) used for this connection.
Start_time	The time stamp on the first packet received for this connection.
Delta_time	The difference between the time stamp of the most recent packet of this connection and the Start_time.
Connection_state	The state of the connection. States for a connection include information such as: NEW-CONNECTION, CONNECTION-IN-PROGRESS, and CONNECTION-CLOSED.
Security_state	The current evaluation of the security state of this connection.
Initiator_pkts	The number of packets the host which initiated the connection has placed on the network.
Initiator_bytes	The number of bytes, excluding protocol headers, contained in the packets.
Receiver_pkts	The number of packets the host which received the connection has placed on the network.
Receiver_bytes	The number of bytes, excluding protocol headers contained in the packets.
Dimension	The dimension of the Initiator_X and the Receiver_X vectors. This value is the number of strings patterns being looked for in the data.
Initiator_X	A vector representing the number of strings matched in Initiator_bytes.
Receiver_X	A vector representing the number of strings matched in Receiver_bytes.

■ Table 1. Connection vector.

Second, audit trails are often not available in a timely fashion. Some IDSs are designed to perform their analysis on a separate host, so the audit logs must be transferred from the source host to a different machine for data analysis [5]. Furthermore, the operating system can often delay the writing of audit logs by several minutes [15]. The broadcast nature of a LAN, however, gives the NSM nearly-instant access to all data as soon as this data is transmitted on the network. It is then possible to immediately start the attack detection process.

Third, the audit trails are often vulnerable. In some past incidents, the intruders have turned off audit daemons or modified the audit trail. This action can either prevent the detection of the intrusion, or it can remove the capability to perform accountability (who turned off the audit daemons?) and damage control (what was seen, modified, or destroyed?) The NSM, on the other hand, passively listens to the network, and is therefore logically protected from subversion. Since the NSM is invisible to the intruder, it cannot be turned off (assuming it is physically secured), and the data it collects cannot be modified.

Fourth, the collection of audit trails degrades

the performance of a machine being monitored (typically between 5 and 20 percent). Unless audit trails are being used for accounting purposes, system administrators often turn off auditing. If analysis of these audit logs is also to be performed on the host, added degradation will occur. If the audit logs are transferred across a network or a communication channel to a separate host for analysis, loss of network bandwidth as well as loss of timeliness of the data will occur. In many environments, the degradation of monitored hosts or the loss of network bandwidth may discourage administrators from using such an IDS. The alternative, namely, the NSM architecture, does not degrade the performance of the hosts being monitored. The monitored hosts are not aware of the NSM, so the effectiveness of the NSM is not dependent on the system administrator's configuration of the monitored hosts.

And, finally, many of the more seriously documented cases of computer intrusions have utilized a network at some point during the intrusion, i.e., the intruder was physically separated from the target. With the continued proliferation of networks and interconnectivity, the use of networks in attacks will only increase. Furthermore, the network itself, being an important component of a computing environment, can be the object of an attack. The NSM can take advantage of the increase of network usage to protect the hosts attached to the networks. It can monitor attacks launched against the network itself, an attack that host-based audit trail analyzers would probably miss.

System Organization (The NSM Model) — The NSM models the network and hosts being monitored in a hierarchically-structured Interconnected Computing Environment Model (ICEM). The ICEM is composed of six layers, the lowest being the bit stream on the network, and the highest being a representation for the state of the entire networked system.

The bottom-most, or first, layer is the packet layer. This layer accepts as input a bit stream from a broadcast LAN, e.g., Ethernet. The bit stream is divided up into complete Ethernet packets, and a time stamp is attached to the packet. This time-augmented packet is then passed up to the second layer. Application of the NSM to other LAN environments is straightforward.

The next layer, called the thread layer, accepts as input the time-augmented packets from the packet layer. These packets are then correlated into unidirectional data streams. Each stream consists of the data (with the different layers of protocol headers removed) being transferred from one host to another host by a particular protocol (e.g., TCP/IP or UDP/IP), through a unique set (for the particular set of hosts and protocol) of ports. This stream of data, called a thread, is mapped into a thread vector. All the thread vectors are passed up to the third layer.

The connection layer, which is the third layer, accepts as input the thread vectors generated by the thread layer. Each thread vector is paired, if possible, to another thread vector to represent a bidirectional stream of data (i.e., a host-to-host connection). These pairs of thread vectors are represented by a connection vector generated by the

combination of the individual thread vectors. Each connection vector is analyzed, and a reduced representation, a reduced connection vector, is passed up to the fourth layer.

Layer 4 is the host layer, which accepts as input the reduced connection vectors generated by the connection layer. The connection vectors are used to build host vectors. Each host vector represents the network activities of a single host. These host vectors are passed up to the fifth layer.

The connected-network layer is the next layer in the ICEM hierarchy. It accepts as input the host vectors generated by the host layer. The host vectors are transformed into a graph *G* by treating the *Data_path_tuples* of the host vectors as an adjacency list. If *G*(host1,host2, serv1) is not empty, then there is a connection, or path, from host1 to host2 by service serv1. The value for location *G*(host1,host2,serv1) is non-empty if the host vector for host1 has (host2,serv1) in its *Data_path_tuples*. This layer can build the connected sub-graphs of *G*, called a connected-network vector, and compare these sub-graphs against historical connected sub-graphs. This layer can also accept questions from the user about the graph. For example, the user may ask if there is some path between two hosts — through any number of intermediate hosts — by a specific service. This set of connected-network vectors is passed up to the sixth and final layer.

The top-most layer, called the system layer, accepts as input the set of connected-network vectors from the connected-network layer. The set of connected-network vectors is used to build a single system vector representing the behavior of the entire system.

System Operation (Detecting Intrusive Behavior)

— The traffic on the network is analyzed by a simple expert system. The types of inputs to the expert system are described below.

The current traffic cast into the ICEM vectors as discussed above is the first type of input. Currently, only the connection vectors and the host vectors are used. The components for these vectors are presented in Tables I and II.

The profiles of expected traffic behavior are the second type of input. The profiles consist of expected data paths (namely, which systems are expected to establish communication paths to which other systems, and by which service?) and service profiles (namely, what is a typical telnet, mail, finger, etc., expected to look like?) Combining profiles and current network traffic gives the NSM the ability to detect anomalous behavior on the network.

The knowledge about capabilities of each of the network services is the third type of input (e.g., telnet provides the user with more capability than FTP does).

The level of authentication required for each of the services is the fourth type of input (e.g., finger requires no authentication, mail requests authentication but does not verify it, and telnet requires verified authentication).

The level of security for each of the machines is the fifth type of input. This can be based on the National Computer Security Center (NCSC) rating of machines, history of past abuses on different machines, rating received after running system evaluation software such as Security Profile Inspector (SPI) [30]

Component	Description
Host_ID	Unique integer used to reference this particular host.
Host_address	The internet address of this host.
Host_state	The state of the host. States include: ACTIVE, NOT_ACTIVE.
Security_state	The current evaluation of the security of this particular host.
Data_path_number	The number of data paths the currently connected to this host. It may be considered the number of arcs from a node in a graph.
Data_path_tuples	A list of four-tuple representing a data path from or to the host. The tuple consists of: Other_host_address, Service_ID, Initiator_tag, and Security_state (of the data path).

■ Table 2. Host vector.

or COPS, or simply which machines the SSO has some control over and which machines the SSO has no control over (e.g., a host from outside the monitored LAN environment would fall in the second category).

The sixth type of input is signatures of past attacks.

The data from these sources is used to identify the likelihood that a particular connection represents intrusive behavior, or if a host has been compromised. The *security_state*, or suspicion level, of a particular connection is a function of four factors: the abnormality of the connection, the security level of the service being used for the connection, the direction of the connection sensitivity level, and the matched signatures of attacks in the data stream for that connection. We elaborate on these components of the *security_state* in the following paragraphs.

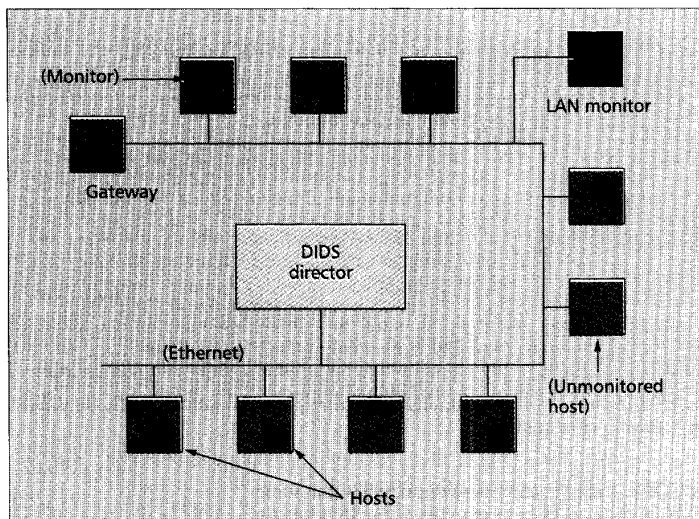
The abnormality of a connection is based on the probability of that particular connection occurring and the behavior of the connection itself. If a connection from host A to host B by service C is rare, then the abnormality of that connection is high. Furthermore, if the profile of that connection compared to a typical connection by the same type of service is unusual (e.g., the number of packets or bytes is unusually high in one direction for a FTP connection), the abnormality of that connection is high.

The security level of the service is based on the capabilities of that service and the authentication required by that service. The TFTP service, for example, has great capabilities with no authentication, so the security level for TFTP is high. The telnet service, on the other hand, also has great capabilities, but it also requires strong authentication. Therefore, the security level for telnet is lower than that of TFTP.

The direction of connection sensitivity level is based on the sensitivity levels of the two machines involved and on which host initiated the connection. If a low-sensitivity-level host connects to or attempts to connect to a high-sensitivity-level host, the direction of connection sensitivity level of that connection is high. On the other hand, if a high-sensitivity-level host connects to a low-level host, the direction of connection security level is low.

The matched signatures of attacks consist of the vectors *Initiator_X* and *Receiver_X* which are simply lists of counts for the number of times some predetermined strings being searched for in the data is matched.

The connection vectors are essentially treated as



■ Figure 1. DIDS target environment

records in a database, and presentation of the information may be made as simple requests into the database. The default presentation format sorts the connections by suspicion level and presents the sorted list from highest suspicion level to the lowest. Presentations can also be made by specifying time windows for connection, connections from a specific host, connections with a particular string matched, etc.

The *security_state*, or suspicion level, of a host is simply the maximum *security_state* of its connection vectors over a particular window of time. The host vectors are also treated as records into a database, and they may be presented in a similar fashion as the connection vectors.

The NSM prototype has been deployed at UC Davis, Lawrence Livermore National Laboratory, and other DOE and Air Force sites. During a particular two-month test period, the NSM analyzed more than 111,000 connections on the Computer Science LAN segment at UC Davis, and it correctly identified more than 300 of these connections as intrusive. These security incidents spanned more than 40 different computers, at least four different hardware platforms, and at least six OS types. The majority of these incidents were associated with attempted break-ins, and some were successful as well. Additional incidents include stealing of password files, running of password crackers on password files, accessing of closed accounts by ex-students, reading of other people's mail, looking around in other people's directories, and using other people's accounts. It should be remarked that approximately only one percent of these security incidents were detected independently by the system administrators.

Distributed Intrusion Detection System (DIDS)

Overview — The Distributed Intrusion Detection System (DIDS) is a joint project between UC Davis, Lawrence Livermore National Laboratory, Haystack Laboratory, and the US Air Force [31, 32]. It is an outgrowth of the NSM project. DIDS is designed to guard against some of NSM's deficiencies, e.g., the NSM cannot monitor an attacker who enters a system via a dial-up line and hence may not generate

any network activity and the NSM can be spoofed via encrypted traffic. Also, one would like to extend the network intrusion detection concept from the LAN environment to arbitrarily wider areas with the network topology being arbitrary as well.

In DIDS, each host in the monitored domain is also equipped with a host monitor. In the current DIDS design, these hosts are assumed to be connected via a LAN, which is monitored by a LAN monitor. Thus, network monitor data is augmented by data from monitored hosts. Generalization of the monitored environment beyond the local area is an open problem, and a preliminary design has been proposed [33].

In DIDS, the host and LAN monitors report any "interesting" events, which may possibly lead to intrusive activity, to a centrally-located DIDS director. The director employs an expert system to detect possible attacks. An initial prototype of DIDS has successfully demonstrated its ability to track users as they move around the network (possibly attempting to hide their true identities) and to identify doorknob rattling attacks [32, 34]. The DIDS architecture performs information aggregation so that even if the activity of a network-wide user may not be suspicious on a single host, the aggregate behavior may be suspect or unpermitted. One of the strengths of this architecture is that it performs "accountability" by tying users with their actions.

System Organization — The generalized distributed environment is heterogeneous, i.e., the network nodes can be hosts or servers from different vendors (Fig. 1). The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDSs. The components of DIDS are the DIDS director, a single host monitor per host, and a single LAN monitor for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation. Reports are sent independently and asynchronously from the host and LAN monitors to the DIDS director through a communications infrastructure (Fig. 2). High-level communication protocols between the components are based on the ISO Common Management Information Protocol (CMIP) recommendations, allowing for future inclusion of CMIP management tools as they become useful. The architecture also provides for bidirectional communication between the DIDS director and any monitor in the configuration. This communication consists primarily of notable events and anomaly reports from the monitors. The director can also make requests for more detailed information from the distributed monitors via a GET directive, and issue commands to have the distributed monitors modify their monitoring capabilities via a SET directive. A large amount of low-level filtering and some analysis is performed by the host monitor to minimize the use of network bandwidth in passing evidence to the director.

The host monitor consists of a host event generator (HEG) and a host agent. The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for notable events,

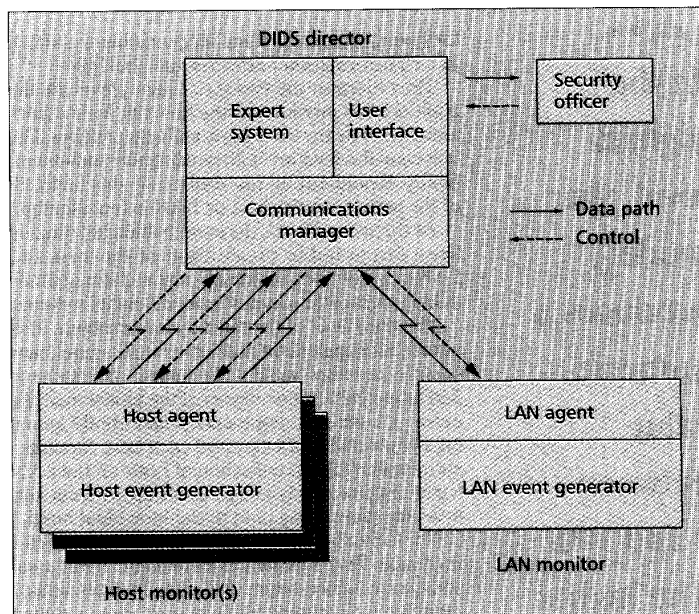
which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as `rlogin` and `rsh`. These notable events are then sent to the director for further analysis. The HEG also tracks user sessions and report anomalous behavior aggregated over time through user and group profiles and the integration of the HAYSTACK intrusion detection algorithm [5] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.

Like the host monitor, the LAN monitor consists of a LAN event generator (LEG) and a LAN agent. The LEG is currently a subset of the NSM [17, 27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on network activities such as `rlogin` and `telnet` connections, the use of security-related services, and changes in network traffic patterns.

The DIDS director consists of three major components that are all located on the same dedicated workstation — a communications manager, an expert system, and a user interface. Because the components are logically independent processes, they could be distributed as well. The communications manager is responsible for the transfer of data between the director and each of the host and the LAN monitors. It accepts the notable event records from each of the host and LAN monitors and sends them to the expert system. On behalf of the expert system or user interface, it is also able to send requests to the host and LAN monitors for more information regarding a particular user. The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole. The expert system is a rule-based system with simple learning capabilities. The director's user interface allows the SSO interactive access to the entire system. The SSO can watch activities on each host, watch network traffic (by setting "wire-taps"), and request more specific types of information from the monitors.

It is anticipated that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS. This will give the SSO the ability to actively respond to attacks against the system in real time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.

System Operation — 1) The Network-user Identification (NID) One of the challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and in tracking an intruder to his source. In a networked



■ Figure 2. Communications architecture.

environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX `who` or `finger` command). However, it may be indicative of an attack (or a preparation of an attack) if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts. Detecting this type of behavior requires attributing multiple sessions, perhaps with different account names, to a single source.

This problem is unique to the network environment and has not been dealt with before in this context. The solution to the multiple user identity problem is to create a network-user identification (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user [34]. All evidence about the behavior of any instance of the user is then accountable to the single NID. In particular, we must be able to determine that `smith@host1` is the same user as `jones@host2`, if in fact they are. Since the NID problem involves the collection and evaluation of data from both the host and LAN monitors, examining it is a useful method to understand the operation of DIDS.

2) The Expert System. DIDS utilizes a rule-based (or production) expert system written in CLIPS. The expert system uses rules derived from a hierarchical Intrusion Detection Model (IDM). The IDM describes the transformation from the distributed raw audit data to high-level hypotheses about intrusions and about the overall security of the monitored environment. This unified view of the distributed system simplifies the recognition of intrusive behavior that spans individual hosts. The model is the basis of the rule-base. The IDM consists of

■ ■ ■ ■ ■

At the highest level, the model produces a numeric value between 1 and 100. The higher the number, the less secure is the network.

six layers, with each layer representing the result of a transformation performed on the data.

The objects at the first (i.e., lowest) level are the audit records provided by the host OS, by the LAN monitor, or by any third-party auditing package. The objects at this level are both syntactically and semantically dependent on the source. At this level, all of the activity on the host or LAN is represented.

At the second level, the *event* (which has already been discussed in the context of the host and LAN monitor) is both syntactically and semantically independent of the source standard format for events.

The third layer of the IDM creates a subject. This introduces a single identification for a user across many hosts on the network. It is the subject who is identified by the NID. Upper layers of the model treat the network-user as a single entity, essentially ignoring the local identification on each host. Similarly, above this level, the collection of hosts on the LAN is generally treated as a single distributed system with little attention being paid to the individual hosts.

The fourth layer of the model introduces the event in context. There are two kinds of context: temporal and spatial. As an example of temporal context, behavior which is unremarkable during standard working hours may be suspicious during off-hours [35]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX who command followed closely by a login or logout is more likely to be related to an intrusion than either of those events occurring alone. Such related patterns of behavior are also referred to as attack signatures [23, 24]. Spatial context implies the relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source. For instance, a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction. The model also allows for the correlation of multiple events from the same user or source. In both of these cases, multiple events are more noteworthy when they have a common element than when they do not.

The fifth layer of the model considers the *threats* to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into *attacks*, *misuses*, and *suspicious acts*. Attacks represent abuses in which the state of the machine is changed. That is, the file system or process state is different after the attack than it was prior to the attack. Misuses represent out-of-policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands which provide information about the state of the system may be suspicious. The targets of abuse are characterized as being either system objects or user objects and as being either

passive or active. User objects are owned by non-privileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, while active objects are essentially running processes.

At the highest level, the model produces a numeric value between one and 100 which represents the overall security state of the network. The higher the number, the less secure is the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, it provides a quick reference point for the SSO. In fact, in the current implementation, no information is lost since the expert system maintains all the evidence used in calculating the security state in its internal database, and the SSO has access to that database.

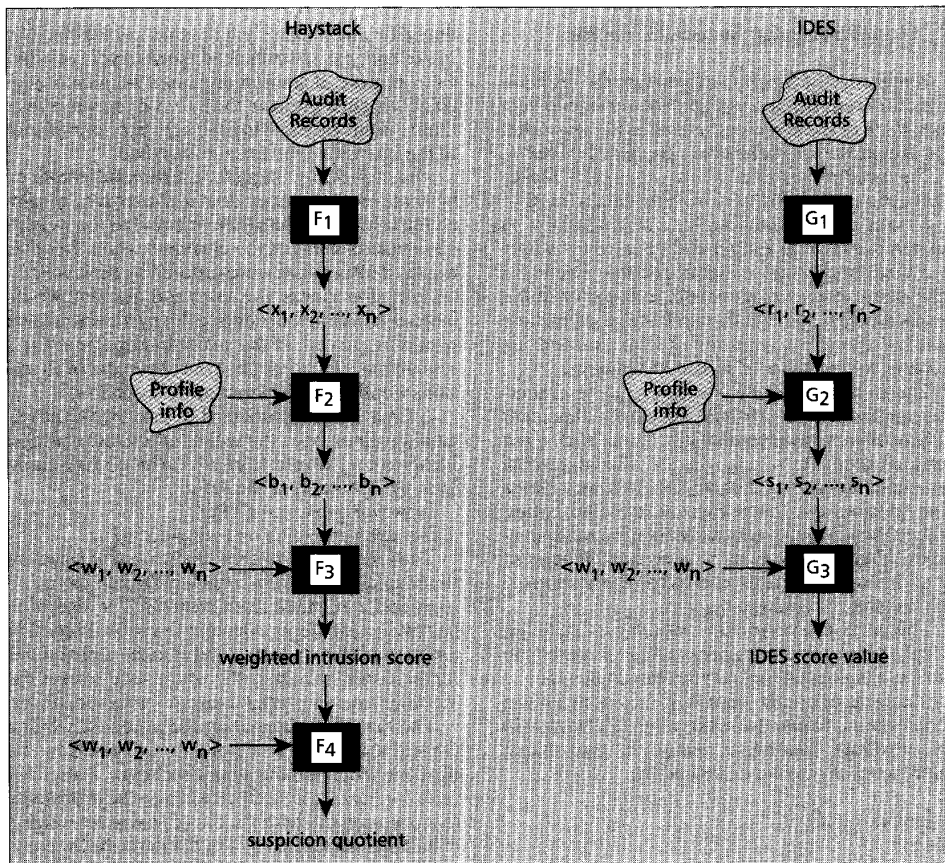
In the context of the NID problem, we are concerned primarily with the lowest three levels of the model: the audit data, the event, and the subject. The generation of the first two of these have already been discussed; the creation of the subject is the focus of the following subsection.

3) Building the NID. With respect to UNIX, the only legitimate ways to create an instance of a user are for the user to login from a terminal, a console, or an off-LAN source, to change the user-id in an existing instance, or to create additional instances (local or remote) from an existing instance. In each case, there is only one initial login (system-wide) from an external device. When this original login is detected, a new unique NID is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new login session, that new session is associated with his original NID. Thus, the system maintains a single identification for each physical user.

We consider an instance of a user to be the 4-tuple `<session_start, user-id, host-id, time>`. Thus each login creates a new instance of a user. In associating a NID with an instance of a user, the expert system first tries to use an existing NID. If no NID can be found which applies to the instance, a new one is created. Trying to find an applicable existing NID consists of several steps. If a user changes identity (e.g., using UNIX's `su` command) on a host, the new instance is assigned the same NID as the previous identity. If a user performs a remote login from one host to another host, the new instance gets the same NID as the source instance. When no applicable NID is found, a new unique NID is created.

The actual association of a NID with a user instance is through the hypothesis `net_user`. A new hypothesis is created for every event reported by the distributed monitors. This new hypothesis, called a subject, is formed by another rule. The rule creates a subject, getting the NID from the `net_user` and the remaining fields from the host audit record, if and only if both the user-id and the host-id match. It is through the use of the subject that the expert system correlates a user's actions regardless of the login name or host-id.

There is still some uncertainty involved with the NID problem. If a user leaves the monitored domain and then comes back in with a different user-id, it is not possible to connect the two instances. Similarly, if a user passes through an unmonitored



■ Figure 3. Statistical intrusion detection algorithms.

host, there is still uncertainty that any connection leaving the host is attributable to any connection entering the host. Multiple connections originating from the same host at approximately the same time also allow uncertainty if the user names do not provide any helpful information. The expert system can make a final decision with additional information from the host and LAN monitors that can (with high probability) disambiguate the connections. To deal with such situations, a new concept, called “thumbprinting,” which is a mechanism for identifying identical network connections, is being developed [33].

Case Study: An Intrusion Detection Algorithm

The intrusion detection algorithm used in the HAYSTACK system is also the core of the host monitor used in DIDS. This algorithm is employed here as an example to demonstrate how a statistical anomaly detection algorithm typically operates.

The HAYSTACK algorithm (hereafter referred to as the algorithm) assumes that the audit trail generated from a host has been converted to a canonical audit trail (CAT) format. Even though different machines may generate audit data in different formats, they can still be served by the same intrusion detection algorithm by having their generated audit information converted to the CAT format. The algorithm

examines a CAT file to generate session vectors representing the activities of the users’ sessions (where a user session includes all activities between a login and a logout). These session vectors are then analyzed against specific types of intrusive activity, and “anomaly scores” for the sessions are calculated. When the scores cross some specific thresholds, warnings reports are generated. The analysis of the session vectors is the specific focus of our discussion here.

A user’s activities are analyzed according to a four-step process. The first step is to generate a session vector representing the activities of the user for a particular session. The second step is the generation of a bernoulli vector representing the attributes which are out of range for a particular session. The third step is the generation of a weighted intrusion score, for a particular intrusion type, from the bernoulli vector and a weighted feature vector. The fourth step is the generation of the suspicion quotient representing how suspicious the algorithm believes this session to be when compared to all other sessions.

This four-step process is shown in Fig. 3. Boxes F_1 through F_4 represent the functions for each of the above processes. Fig. 3 also shows the steps G_1 through G_3 followed by the IDES statistically anomaly detection algorithm [11], and it indicates how the steps in the Haystack and the IDES algorithms parallel each other. We restrict our attention to the Haystack algorithm here. F_1 , the generation of the session vector, is not covered. Functions F_2 through F_4 are discussed below.

.....
The HAYSTACK algorithm assumes that the audit trail generated from a host has been converted to a canonical audit trail (CAT) format.

■ ■ ■ ■ ■

Intrusion detection is a viable and practical approach for providing a different notion of security in our huge and existing infrastructure of computer and network systems.

F₂: Generating the Bernoulli Vector

F_2 represents the function to generate the bernoulli vector from the session and the threshold vectors. Each of these vectors, as well as the algorithm to calculate the bernoulli vector, are described below.

The session vector $X = \langle x_1, x_2, \dots, x_n \rangle$ represents the counts for various attributes used to represent a user's activities for a single session. A session begins with a login audit record for a user and terminates with a logout record. The login and logout times are also included as part of the session vector. Some of the attributes used include session duration, number of files opened for reading, and number of secure I/O failures.

The threshold vector $T = \langle t_1, t_2, \dots, t_n \rangle$ represents the ranges for each attribute i in which 90 percent of historical measurements fall. Thus, each t_i is a tuple of the form $\langle t_{i,min}, t_{i,max} \rangle$. The algorithm assumes that each of the t_i vectors follows a Gaussian distribution. In the current implementation of the algorithm, all measurements are actually one-tailed. That is, $t_{i,min}$ is set to 0, so that 90 percent of all historical measurements for the attribute i are less than or equal to $t_{i,max}$.

The bernoulli vector $B = \langle b_1, b_2, \dots, b_n \rangle$ is a simple binary vector representing which of the attribute counts fall outside the thresholds for the particular user group. That is, if the count for the i^{th} attribute x_i falls outside the range t_i , then b_i will be set to one; otherwise, b_i is set to zero. Thus, the function F_2 can be described by:

```
for i ← 1 to n do
    
$$b_i = \begin{cases} 0 & t_{i,min} \leq x_i \leq t_{i,max} \\ 1 & \text{otherwise} \end{cases}$$

```

F₃: Generating the Weighted Intrusion Score

F_3 represents the function to generate the weighted intrusion score for a particular session and for a particular intrusion type. By itself, the weighted intrusion score is meaningless; however, with the knowledge of the distribution of the weighted intrusion scores for all sessions, this weighted intrusion score can be used to assign a suspicion value to the session. The weighted feature vector and the algorithm to calculate the weighted intrusion score are described below.

A weighted intrusion vector $W = \langle w_1, w_2, \dots, w_n \rangle$ exists for each group and intrusion type pair. Each w_i relates the importance of the i^{th} attribute to detecting the particular intrusion type. Therefore, if $w_i > w_j$, then the fact that the i^{th} attribute exceeds the threshold t_i is more useful in detecting the particular intrusion than the fact that the j^{th} attribute exceeds the threshold t_j .

The weighted intrusion score is simply the sum of all weights, w_i , where the i^{th} attribute exceeded its threshold t_i . Its value is given by:

$$\text{weighted intrusion score} = \sum_{i=1}^n b_i * w_i$$

F₄: Generating the Suspicion Quotient

F_4 represents the function to calculate the suspicion quotient, or suspicion value, for a particular session and intrusion type. The suspicion quotient

describes how closely a session resembles the intrusion type as compared to all other sessions. For example, if session one has a suspicion quotient which is greater than that of session two, then session one is said to have a greater resemblance to the intrusion type than session two.

The suspicion quotient for a session is determined by what percentage of random sessions have a weighted intrusion score less than or equal to the weighted intrusion score of the current session. In other words, if all sessions were ordered by their weighted intrusion scores, the suspicion quotient would indicate where in the ordering this particular session would be placed. If the suspicion quotient for a session was 97 percent, then only 3 percent of all sessions would be considered more suspicious.

A table can be generated to provide the suspicion quotient from the weighted intrusion score. Function F_4 would then be reduced to a table look-up operation of the form:

```
suspicion quotient = TABLE
[weighted_intrusion_score]
```

The following algorithm can be used to calculate the suspicion quotient from a weighted intrusion score and a weighted feature vector. The algorithm has two major steps: calculate the probability that a session will have a score j , and sum the probabilities for all scores $j \leq \text{max_score}$, where max_score is the maximum score any weighted intrusion score can have.

The first step, calculating the probability that a session will have a score of j , is performed by the dynamic programming algorithm below. It uses the table P where $P_{i,j}$ is interpreted to be the probability that a session will have a score of j using the weights of the first i attributes. $Pr_i(0)$ is defined to be the probability that the i^{th} attribute will be within its threshold (and thus $b_i * w_i = 0$). Similarly, $Pr_i(1)$ is the probability that the i^{th} attribute will be outside its threshold (and thus $b_i * w_i = w_i$), i.e., $Pr_i(1) = 1 - Pr_i(0)$. The parameters $\{Pr_i(0), i = 0, 1, \dots, n\}$ are user selectable, and in the HAYSTACK system, they are all set to 0.9. The first step of the algorithm follows.

```
/* initialize the probability table */
 $P_{0,0} = 1.0$ 
for j ← 1 to max_score do
     $P_{0,j} = 0.0$ 
/* fill in the probability table */
for i ← 1 to n do
    for j ← 0 to max_score do
        
$$P_{i,j} = P_{i-1,j} * Pr_i(0) + P_{i-1,j-w_i} * Pr_i(1)$$

```

After the algorithm terminates, the table $P_{n,j}$ describes the probability that a random session will have a weighted intrusion score = j .

The second step of the algorithm calculates the probability that a random session's weighted intrusion score is less than or equal to the weighted intrusion score given. This is given by the equation

$$\text{suspicion quotient} = \sum_{j=0}^{\text{score}} P_{n,j}$$

In summary, the intrusion detection algorithm outlined above analyzes a session vector in three steps: 1) it calculates a bernoulli vector, 2) it calculates the weighted intrusion score, and 3) it calculates the suspicion quotient.

Conclusion

Intrusion detection is a viable and practical approach for providing a different notion of security in our huge and existing infrastructure of (possible insecure) computer and network systems. Intrusion detection systems are based on host-audit-trail and network traffic analysis, and their goal is to detect attacks, preferably in real time. A number of prototype intrusion detection systems have been built, and this concept has been proven to be extremely promising.

In the future, it is expected that the current prototypes will be developed further in order to turn them into production-quality systems. Benchmarking mechanisms in order to test the effectiveness of IDSs should be developed. Accurate approaches for representing attacks and misuse (including development of models for new attack methods) as well as new and more effective detection strategies must be investigated. In addition, much more research is expected to be conducted, e.g., how can the intrusion-detection concept be extended to arbitrarily large networks (e.g., the worldwide Internet), how can the IDS itself be protected from attackers, etc.

Acknowledgements

The intrusion detection R & D work in the Computer Security Laboratory at UC Davis is supported by the Lawrence Livermore National Laboratory (LLNL), U.S. Air Force Cryptologic Support Center (AFCSC), National Computer Security Center, Hewlett-Packard, Logicon-Ultrasonics, and the State of California MICRO Program.

We acknowledge the contributions of our colleagues at UC Davis, LLNL, AFCSC, and Haystack Laboratories toward the NSM and DIDS prototypes. It is through various discussions with them as well as with people at NCSC (Becky Bace in particular) that our understanding of intrusion detection has improved.

References

- [1] National Computer Security Center, Department of Defense, Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, Dec. 1985 (Orange Book).
- [2] National Computer Security Center, Dept. of Defense, Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, July 1987 (Red Book).
- [3] V. L. Vodyck and S. T. Kent, "Security in high-level network protocols," *IEEE Commun. Mag.*, vol. 23, no. 7, pp. 12-24, July 1985.
- [4] D. E. Denning, "An intrusion-detection model," *IEEE Trans. on Software Engg.*, vol. SE-13, pp. 222-232, Feb. 1987.
- [5] S. E. Smaha, "Haystack: An Intrusion Detection System," Proc., IEEE Fourth Aerospace Computer Security Applications Conference, Orlando, FL, Dec. 1988.
- [6] C. Dowell and P. Ramstedt, "The COMPUTERWATCH data reduction tool," Proc., 13th National Computer Security Conference, Washington, DC, pp. 99-108, Oct. 1990.
- [7] W. T. Tener, "Discovery: an expert system in the commercial data security environment," Proc. Fourth IFIP TC11 International Conference on Computer Security, North-Holland, Dec. 1986.
- [8] W. T. Tener, "AI & 4GL: Automated Detection and Investigation Tools," Proc. Fifth IFIP International Conference on Computer Security, North-Holland, May 1988.
- [9] T. F. Lunt et al., "IDES: A Progress Report," Proc., Sixth Annual Computer Security Applications Conf., Tucson, AZ, Dec. 1990.
- [10] T. F. Lunt et al., "A Real-time Intrusion Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.
- [11] H. S. Javitz and A. Valdez, "The SRI IDES Statistical Anomaly Detector," Proc., 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1991.
- [12] J. R. Winkler and W. J. Page, "Intrusion and Anomaly Detection in Trusted Systems," Proc., Fifth Annual Computer Security Applications Conference, Tucson, AZ, Dec. 1989.
- [13] J. R. Winkler, "A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks," Proc. 13th National Computer Security Conference, Washington, D.C., Oct. 1990, 115-124.
- [14] M. M. Sebring et al., "Expert systems in intrusion detection: A case study," Proc., 11th National Computer Security Conf., Baltimore, MD, Oct. 1988.

- [15] H. S. Vaccaro and G. E. Liepins, "Detection of anomalous computer session activity," Proc., 1989 Symposium on Research in Security and Privacy, Oakland, CA, pp. 280-289, May 1989.
- [16] J. Hochberg et al., "NADIR: an automated system for detecting network intrusion and misuse," *Computers and Security*, vol. 12, no. 3, pp. 235-248, May 1993.
- [17] L. T. Heberlein et al., "A network security monitor," Proc., 1990 Symposium on Research in Security and Privacy, Oakland, CA, pp. 296-304, May 1990.
- [18] T. F. Lunt, personal communication, 1992.
- [19] S. M. Bellovin, "There Be Dragons," Proc., Third UNIX Security Symposium, Baltimore, MD, Sept. 1992.
- [20] W. R. Cheswick, "An evening with berferd, in which a cracker is lured, endured, and studied," Proc., Winter USENIX Conference, San Francisco, Jan. 1992.
- [21] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *ACM Computer Commun. Review*, vol. 19, no. 2, pp. 32-48, April 1989.
- [22] H. S. Teng, K. Chen, and S. C.-Y. Lu, "Adaptive real-time anomaly detection using inductively generated sequential patterns," Proc., 1990 Symposium on Research in Security and Privacy, Oakland, CA, May 1990.
- [23] S. R. Snapp, B. Mukherjee, and K. N. Levitt, "Detecting intrusions through attack signature analysis," Proc., 3rd Workshop on Computer Security Incident Handling, Herndon, VA, Aug. 1991.
- [24] S. R. Snapp, Signature Analysis and Communication Issues in a Distributed Intrusion Detection System, M.S. thesis, Division of Computer Science, University of California, Davis, Aug. 1991.
- [25] J. Doak, Intrusion Detection: The Application of Feature Selection, a Comparison of Algorithms, and the Application of a Wide Area Network Analyzer, M.S. thesis, Division of Computer Science, University of California, Davis, August 1992.
- [26] T. Burr, et al., "Software toolkit for analysis research (STAR)," Technical Report No. LA-12617-MS, Los Alamos National Laboratory, Aug. 1993.
- [27] L. T. Heberlein, Towards Detecting Intrusions in a Networked Environment, M. S. thesis, Division of Computer Science, University of California, Davis, June 1991.
- [28] L. T. Heberlein et al., "Towards detecting intrusions in a networked environment," Proc., 14th DOE Conference on Computer Security, Concord, CA, May 1991, pp. 17-47 — 17-65.
- [29] L. T. Heberlein, K. N. Levitt, and B. Mukherjee, "A method to detect intrusive activity in a networked environment," Proc., 14th National Computer Security Conference, Washington, DC, pp. 362-371, Oct. 1991.
- [30] T. Bartoletti, "SPI/UNIX: Security Profile Inspector for UNIX computer systems," Proc., 3rd Workshop on Computer Security Incident Handling, Herndon, VA, Aug. 1991.
- [31] S. R. Snapp et al., "A system for distributed intrusion detection," Proc., IEEE COMPCON 91, San Francisco, CA, Feb. 1991, pp. 170-176.
- [32] S. R. Snapp et al., "DIDS (Distributed Intrusion Detection System) — Motivation, Architecture, and An Early Prototype," Proc., 14th National Computer Security Conf., Washington, D.C., Oct. 1991.
- [33] L. T. Heberlein, B. Mukherjee, and K. N. Levitt, "Internet security monitor: An intrusion-detection system for large-scale networks," Proc., 15th National Computer Security Conference, Baltimore, MD, Oct. 1992.
- [34] C. Ko et al., "Analysis of an algorithm for distributed recognition and accountability," Proc., First ACM Conf. on Computer and Communications Security, Fairfax, VA, pp. 154-164, Nov. 1993.
- [35] T. F. Lunt, "Automated audit trail analysis and intrusion detection: A survey," Proc., 11th National Computer Security Conf., Baltimore, MD, Oct. 1988.
- [36] J. Brentano, An Expert System for Detecting Attacks on Distributed Computer Systems, M. S. Thesis, Division of Computer Science, University of California, Davis, Feb. 1991.

Biographies

BISWANATH MUKHERJEE [M'87] received his B. Tech. degree from Indian Institute of Technology, Kharagpur, India, in 1980 and the Ph.D. degree from University of Washington, Seattle, in 1987, where he held a GTE Teaching Fellowship and a General Electric Foundation Fellowship. He is currently at the University of California, Davis, where he has been an Associate Professor of Computer Science since 1992. He is co-winner of the 1991 National Computer Security Conference Outstanding Paper Award. His research interests include lightweight networks and network security.

L. TODD HEBERLEIN is a postgraduate researcher in the computer Science Department at UC Davis. He received his M.S. in computer science from UC Davis in 1991. He was the chief architect and implementor of the UC Davis Network Security Monitor (NSM), and he has been a principal developer of the US Air Force's Distributed Intrusion Detection System (DIDS). He has authored or co-authored ten papers in intrusion detection, one receiving an outstanding paper award, and he was co-editor for the Proceedings of the Workshop on Future Directions in Computer Misuse and Anomaly Detection.

KARL N. LEVITT is a professor of computer science at UC Davis. He came to Davis in March 1986 after having worked at SRI International for 20 years, for five of which he served as director of the Computer Science Laboratory. He received his Ph.D. in electrical engineering from New York University in 1966, where his research was concerned with Information Theory and Error Correcting Codes. His interests include automated verification, computer security, fault-tolerant computing, advanced architectures and software engineering. Papers he co-authored received Best Paper Awards at conferences in 1968 and 1991. He was co-chair of the HOL conference and an intrusion detection workshop at UC Davis in 1991.

■ ■ ■ ■ ■
Accurate approaches for representing attacks and misuse as well as new and more effective detection strategies must be investigated.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.

v.

TAASERA LICENSING LLC

Petition for *Inter Partes* Review of

U.S. Patent No. 9,092,616

DECLARATION OF INGRID HSIEH-YEE, PH.D.

TABLE OF CONTENTS

I.	Introduction	3
A.	Qualifications and Professional Experience	3
B.	Scope of This Declaration.....	11
C.	Person of Ordinary Skills in the Art	12
D.	Evidence Considered in Forming My Opinions	13
II.	Authenticity and Public Availability of <i>Mukherjee</i> (EX1005)	14
A.	Authentication.....	14
B.	Library of Congress Copy.....	15
C.	Library of Congress Records	17
D.	Copyright Registration Record	20
E.	Linda Hall Library Copy.....	21
F.	Linda Hall Library Records	22
G.	Usage Record.....	24
H.	Summary of My Opinion on <i>Mukherjee</i>	24
III.	Conclusion.....	25

I. Introduction

I, Ingrid Hsieh-Yee, Ph.D., do hereby declare as follows:

1. I have personal knowledge of all of the matters about which I testify in this declaration.

2. I have been retained as an independent expert witness on behalf of Trend Micro, Inc. (“TREND”) for proceedings before the United States Patent and Trademark Office (“PTO”) regarding the authenticity and public availability of certain prior art reference in relation to US Patent No. 9,092,616 (“the ’616 patent”).

3. I am being compensated for my work in this matter at my customary hourly rate. I am also being reimbursed for any reasonable expenses associated with my work and testimony in this investigation. My compensation is not contingent on the results of my study, the substance of my opinions, or the outcome of this matter.

A. Qualifications and Professional Experience

4. My complete qualifications and professional experience are described in my academic curriculum vitae (**Appendix A**). The following is a brief summary of my relevant qualifications and professional experience.

5. In 2022 I was conferred the title of Professor Emerita after 32 years as a professor in the Department of Library and Information Science of the Catholic University of America. I have experience working in an academic library, a medical library, and a legislative library. I hold a Ph.D. in Library and Information Studies

from the University of Wisconsin-Madison and a Masters in Library and Information Studies from the University of Wisconsin-Madison.

6. I am an expert on library cataloging and classification and have published two editions on this subject, *Organizing Audiovisual and Electronic Resources for Access: A Cataloging Guide* (2000 and 2006). I taught a variety of courses, including Cataloging and Classification, Advanced Cataloging and Classification, Organization of Internet Resources, Organization of Information, Digital Content Creation and Management, Metadata, Internet Searches and Web Design, Information Literacy Instruction, Advanced Information Retrieval and Analysis Strategies, and The Information Professions in Society. I am familiar with metadata schema design and implementation. In my teaching I covered the design and implementation of metadata in libraries, databases, search engines, digital repositories, digital libraries, and digital archives. I also covered how information organization affects the discovery and access to digital resources on the Internet. My research interests cover cataloging and classification, information organization, metadata, information retrieval, information architecture, digital collections, scholarly communication, social media, user interaction with information systems, and others.

7. I am very familiar with a library cataloging encoding standard known as the “Machine-Readable Cataloging” standard, also known as “MARC,” which

became the national standard for sharing bibliographic data in the United States by 1971 and the international standard by 1973. MARC is the primary communications protocol for the transfer and storage of bibliographic metadata in libraries. Experts in my field reasonably rely upon MARC records when forming their opinions.

8. A MARC record consists of several fields, each of which contains specific data about the work. Each field is identified by a standardized, unique, three-digit code corresponding to the type of data that follows. **Appendix B** is a true and correct copy of Parts VII to X of “Understanding MARC Bibliographic: Machine-Readable Cataloging” (<http://www.loc.gov/marc/umb/um07to10.html>) from Library of Congress. *Understanding MARC Bibliographic* was originally published by the Follett Software Co. and copyrighted in 1988. Editions in 1989, 1990, 1994, 1998, 2000, 2003 and 2009 include changes. The latest edition was copyrighted in 2009. This document explains commonly-used MARC fields. For example, the personal author of the work is recorded in Field 100, the title is recorded in Field 245, publisher information is recorded in Field 260, the physical volume and characteristics of a publication are recorded in Field 300, and topical subjects are recorded in the 650 fields.

9. The Online Computer Library Center (OCLC) is the largest bibliographic network of the world, with more than 554 million records and thousands of member institutions (many of which are libraries of some type) in more

than 100 countries. OCLC was founded in 1967 to promote and support library cooperation. According to the “Third Article, Amended Articles of Incorporation of OCLC Online Computer Library Center, Inc.,” OCLC was created “to establish, maintain and operate a computerized library network and to promote the evolution of library use, of libraries themselves, and of librarianship, and to provide processes and products for the benefit of library users and libraries, including such objectives as increasing availability of library resources to individual library patrons and reducing the rate of rise of library per-unit costs, all for the fundamental public purpose of furthering ease of access to and use of the ever-expanding body of worldwide scientific, literary and educational knowledge and information” (source: <https://www.oclc.org/content/dam/oclc/membership/articles-of-incorporation.pdf>).

10. OCLC members can contribute original cataloging records in MARC to the system or derive cataloging records from existing records, an activity referred to as “copy cataloging.” When an OCLC participating institution acquires a work, it can create an original MARC record for the work in OCLC’s Connexion system (a system for catalogers to create and share MARC records), and the system will automatically generate a code for the date of record creation in the yymmdd format, and the creating library’s OCLC symbol is recorded in subfield “a” of the 040 field. Once the MARC record is in Connexion, it becomes available to other OCLC members for adoption to their local online catalogs (*i.e.*, copy cataloging).

11. After a MARC record is created in Connexion, it also becomes searchable and viewable on WorldCat, which is a free web portal for users to explore more than 10,000 libraries worldwide. The record in WorldCat, however, is not presented in MARC fields. Instead, the data elements are labeled to help users interpret the record. Thus, the information stored in MARC records in Connexion is available to the interested public through the user-friendly WorldCat web portal.

12. WorldCat (<http://www.worldcat.org>) is “the world’s largest network of library content and services” and its features are summarized in “About WorldCat” (<https://www.worldcat.org/about>). WorldCat began operation in January 1998 as a free web portal to more than 10,000 library collections worldwide. Through WorldCat, users can search for information resources in their local libraries and libraries around the world. WorldCat allows users to search for books, CDs, videos, and many new types of digital content, such as audiobooks, in many languages. Users can also retrieve research materials and article citations with links to their full text. After an item is retrieved, WorldCat helps users identify a library nearby that holds the item or all the libraries that hold the item. WorldCat is an efficient way to explore the content held by more than 10,000 libraries around the world. WorldCat is publicly accessible, and there is no fee to perform a search on WorldCat.

13. Library online catalogs gained acceptance in the early 1980s and many libraries migrated their systems to the World Wide Web in the mid-1990s. Library

online catalogs are based on MARC records that represent their collections in order to help the public understand what materials are publicly accessible in those libraries. Most libraries with online catalogs have made their catalogs freely available on the Web. These online catalogs offer user-friendly search interfaces. Strong user interest in keyword searches and the popularity of Google have led to the “googlization” of library search systems. As a result, many library catalogs now provide a single search box for users to conduct keyword searches, with additional support for searches by author, title, subject terms, and other data elements such as ISBN (International Standard Book Number). Library catalogs these days also offer features for users to narrow their search results by language, year, format, and other elements. Many libraries display MARC records on their online catalogs with labels for the data elements to help the public interpret MARC records. Many libraries also offer the option to display MARC records in MARC fields.

14. Libraries create MARC records for works they acquire, including books, serials, motion pictures, and publications in other formats. Monograph cataloging is fairly common in libraries, and most libraries make a newly cataloged monograph available to the public soon after the cataloging work is completed, usually within a week. Libraries can create original cataloging records or use an existing record in OCLC to create a copy cataloging record. As soon as the cataloging record is completed, it is added to the library’s online catalog for users.

If the record is an original record, it is also entered into the OCLC system. If it is a copy cataloging record, the library's holding symbol is attached to the existing original record in the OCLC system to facilitate searching and interlibrary loan.

15. The cataloging of serials and the serial check-in process are discussed here to show how libraries usually provide access to newly received serial issues. According to the glossary of the *RDA: Resource Description and Access* cataloging standard, a serial is “a mode of issuance of a manifestation issued in successive parts, usually bearing numbering, that has no predetermined conclusion. A serial includes a periodical, monographic series, newspaper, etc.” Because the publisher of a serial makes new issues of the serial available successively, a customary cataloging practice is to create one bibliographic record for the serial, and the serial record (encoded in MARC) typically provides information on the beginning date and frequency of the serial, not the dates of individual issues. In other words, libraries typically do not create MARC records for individual issues of a serial. Instead, they rely on a serial check-in system to track the receipt of new issues. A common check-in practice is to date stamp a new issue when it arrives. This practice has become automated since the late 1990s, and libraries now vary in how they share the receipt date of a new serial issue with the public. Some libraries use a date stamp, some affix a label to indicate the receipt date, some pencil in the receipt date, and some do not provide the information to the public.

16. The serial check-in process usually takes less than an hour, and one of the steps involves placing a date stamp on the new issue to document the date the issue is checked in. After that, the holdings information of the serial is updated in the library's catalog so that users know which issues are available for request or access. After serial check-in is completed, the new issue is placed on the shelf with the previous issues of the serial. Libraries with a public periodical room typically place new issues in the periodical room for easy user access. Because information presented in serials often reflects latest discovery, a general practice of libraries is to make new issues of serials available for user access soon after they are checked in, usually within a week.

17. I am personally familiar with many online catalogs, databases, and search engines. In preparing for this declaration, I used the following authoritative information systems to search for records:

- Google Scholar (<https://scholar.google.com>)
- IEEE Xplore Digital Library (<https://ieeexplore.ieee.org/Xplore/home.jsp>)
- Online catalog of the Library of Congress (<https://catalog.loc.gov/>)
- Online catalog of the Linda Hall Library (https://catalog.lindahall.org/discovery/search?vid=01LINDAHALL_INS_T:LHL)

- Public catalog of the US Copyright Office (<https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?DB=local&PAGE=First>)
- WorldCat (<https://www.worldcat.org>)

B. Scope of This Declaration

18. I have been asked to offer opinions on the authenticity and public availability date of the following document:

Mukherjee, B., Heberlein, L. T., & Levitt, K. N., (1994, May),
 “Network intrusion detection,” *IEEE Network*, vol. 8, no. 3, pp. 26-41,
 IEEE, **EX1005** (“*Mukherjee*”).

19. I am rendering my expert opinion on the authenticity of the document referenced herein and on whether it was published. I will provide my opinion on when and how this document was publicly disseminated or otherwise made publicly available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located and retrieved this document.

20. I am informed by counsel that an item is considered authentic if there is sufficient evidence to support a finding that the item is what it is claimed to be. I am also informed that authenticity can be established based on the contents of the documents themselves, such as the appearance, content, substance, internal patterns, protocols followed or other distinctive characteristics of the item.

21. I am informed by counsel that a given reference is “published” if it is publicly accessible upon a satisfactory showing that such a document has been disseminated or otherwise made publicly available to the extent that persons interested and ordinarily skilled in the subject matter or art could locate it exercising reasonable diligence and obtain the document. I have also been informed by counsel that materials available in a library constitute “printed publications” if they are cataloged and indexed according to general library practices and protocols that make the references available and accessible to members of the interested public.

22. I have been informed by counsel that the ’616 patent was filed in the United States on July 27, 2012, and that it claims priority to the provisional application filed on May 1, 2012. I have assessed public accessibility of the documents referenced below as of this claimed priority date.

C. Person of Ordinary Skills in the Art

23. I am told by counsel that the subject matter of this article relates to dynamic operational integrity attestation of application security and a user reputation at runtime.

24. I have been informed by counsel that a “person having ordinary skill in the art at the time of the inventions” (PHOSITA) is a hypothetical person who is presumed to be familiar with the relevant field and its literature at the time of the inventions. This hypothetical person is also a person of ordinary creativity, capable

of understanding the scientific principles and literature applicable to the pertinent field.

25. I am told by counsel that persons of ordinary skill in this subject matter or art would have had a bachelor's degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security topics such as malware/exploit detection and/or network forensics. More education can supplement practical experience and vice versa.

26. It is my opinion that such a person would have been actively engaged in learning about the field, possibly through formal instruction through the bibliographic resources. By at least May 1, 2012, such a person would have had access to a vast array of print and electronic resources, including at least the documents referenced below.

D. Evidence Considered in Forming My Opinions

27. In the preparation of this declaration, I have reviewed the documents referenced below and any other documents I reference herein, and each of these is a type of material that experts in my field would reasonably rely upon when forming their opinions:

- (1) The document referenced above in Section I.B;

- (2) Copy of *Mukherjee* obtained from Library of Congress, **Appendix 1005A**;
- (3) Bibliographic and MARC records for *IEEE Network* that contains *Mukherjee*, available at <https://lccn.loc.gov/87649810> from the online catalog of the Library of Congress, accessed and obtained on February 9, 2023, **Appendix 1005B**;
- (4) Copyright registration record for vol. 8, no. 3 of *IEEE Network* that contains *Mukherjee*, available at https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?v1=22&ti=1,22&Search%5FArg=IEEE%20Network&Search%5FCode=TALL&CNT=25&PID=mXdVOrVd9f5D7qeI9kav_rt2HYiK&SEQ=20230209160558&SID=1 from the public catalog of the US Copyright Office, accessed and obtained on February 9, 2023, **Appendix 1005C**;
- (5) Copy of *Mukherjee* obtained from Linda Hall Library, **Appendix 1005D**;
- (6) Bibliographic and MARC records for *IEEE Network* that contains *Mukherjee*, available at https://catalog.lindahall.org/permalink/01LINDAHALL_INST/191da7s/alma992718683405961 from the online catalog of the Linda Hall Library, accessed and obtained on February 9, 2023, **Appendix 1005E**;
- (7) Early citations to *Mukherjee*, obtained from Google Scholar, **Appendix 1005F**.

These records are identified and discussed in this declaration. Experts in the field would reasonably rely on the data described herein to form their opinions.

II. Authenticity and Public Availability of *Mukherjee* (EX1005)

A. Authentication

28. **EX1005** is a true and correct copy of “Network intrusion detection,” (“*Mukherjee*”), by Mukherjee et al., *IEEE Network*, vol. 8, no. 3, pp. 26-

41, that I obtained from IEEE. When I began preparing this declaration, I searched Google Scholar for *Mukherjee* for records, and a link led me to an IEEE metadata record for this article at <https://ieeexplore.ieee.org/document/283931> that shows *Mukherjee* was published in “May-June 1994.” I then followed the PDF link in this record to purchase this article. This IEEE copy is presented as **EX1005** in this declaration.

29. **EX1005** is a true and correct copy of *Mukherjee* obtained from IEEE. Page 1 (internal page 26) shows the title, abstract, authors, and full text of this article. The bottom of this page shows this article has a copyright date of “1994” and IEEE is the copyright holder. It also shows this article is included in the May/June 1994 issue of *IEEE Network* and the ISSN of the journal is “0890-8044.” **EX1005** shows *Mukherjee* has a total of 16 pages, including two tables, three figures, and 36 references.

B. Library of Congress Copy

30. To determine the availability of *Mukherjee* in libraries, I searched WorldCat by the journal title and found a record at <https://www.worldcat.org/title/14393593> that showed Library of Congress was one of the holding libraries, so I searched their online catalog and found a record at <https://lcn.loc.gov/87649810> that confirmed the holdings information. I then requested a scanned copy of *Mukherjee* via WTS (Wisconsin TechSearch), a

document delivery service based at the University of Wisconsin-Madison. This copy is presented as **Appendix 1005A** in this declaration.

31. **Appendix 1005A** is a true and correct copy of *Mukherjee* obtained from Library of Congress. Page 1 shows the covers and spine of a bound volume and the spine shows the volume contains *IEEE Network*, volume 7-8, 1993-94, and its call number is “TK 5105.5 .I324.” Page 2 is the issue cover that identifies this work as “May/June 1994 Vol. 8 No. 3” of *IEEE Network*, which is a publication of the IEEE Communications Society. This cover carries a date stamp of “Copyright Off... MAY 19 1994 Library of Congress” and an address label showing Library of Congress as the recipient of this issue. Page 3 is the table of contents that shows “[t]his issue features two articles that cover security considerations in data network” and “Network intrusion detection” by Mukherjee et al. begins on page 26. Page 4 is the copyright page that shows “IEEE NETWORK ISSN 0890-8044 is published bimonthly by the Institute of Electrical and Electronics Engineers, Inc.” and this issue has a “1994” copyright date, with the Institute of Electrical and Electronics Engineers, Inc. as the copyright holder. Pages 5 to 20 (internal pages 26 to 41) are the full text of *Mukherjee*. I have closely compared the Library of Congress copy of *Mukherjee* (**Appendix 1005A**) with the publisher copy (**EX1005**), and concluded that these two copies have the same content. The only difference is the tables and figures in the library copy are in color.

32. The Library of Congress copy of *Mukherjee* (**Appendix 1005A**) is in a condition that creates no suspicion about its authenticity. Specifically, this copy is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. In addition, I found this copy to have the same content as the publisher copy (**EX1005**). Moreover, **Appendix 1005A** was found in the custody of a national library, a place where, if authentic, it would likely be found. Accordingly, I see no reason to question the authenticity of this library copy of *Mukherjee* (**Appendix 1005A**).

C. Library of Congress Records

33. **Appendix 1005B** is a true and correct copy of the bibliographic and MARC records for *IEEE Network* that contains *Mukherjee*. I personally located, identified, and obtained these records from the online catalog of the Library of Congress. These are the type of record experts in my field would reasonably rely upon when forming their opinions.

34. Pages 1 to 3 of **Appendix 1005B** are the bibliographic record that identifies this publication as a periodical and shows the title, publisher, publication history, physical description, publication frequency, the journal's relation to the IEEE Communications Society, ISSN and other details. The title, publisher, note on the parent organization of the journal and ISSN match the information presented in the Library of Congress copy (**Appendix 1005A**) confirming this record represents

the journal. The “Item Availability” area on page 3 shows the call number of this journal is “TK5105.5 .I324” and the issues can be requested at the Jefferson or Adams Building Reading Rooms. The list of “Older receipts” shows “v.3-v.17 (1989-2003)” are held by the library, meaning vol. 8, no. 3 that contains *Mukherjee* is available at the library.

35. Pages 4 and 6 are the MARC record. The first six digits of Field 008 show the record creation date was “861014” (*i.e.*, October 14, 1986), and the code of “c19879999” following the record creation date indicates this publication is a continuing resource that began publication in 1987 and has no end date of publication yet. Subfields “a” and “c” of Field 040 show “NSD” created the original MARC record and subfield “d” shows “DLC” modified the record later. “NSD” is the National Serial Data Program of the Library of Congress. According to the *Directory of OCLC Members* (<https://www.oclc.org/en/contacts/libraries.html>), “DLC” is the OCLC library symbol for Library of Congress. Data in Field 008 and 040 inform my opinion that the Library of Congress MARC record was based on the record created by NSD on October 14, 1986.

36. Field 022 shows the ISSN of this journal, Field 245 shows the main title, Fields 246 show three variant titles are included to help users discover this journal, Field 260 shows the publisher and publication year, Field 362 shows the journal began with vol. 1, no. 1 in January 1987, and Field 310 shows the publication

frequency has been “Bimonthly” since 1988. Fields 710 show that the IEEE Communications Society and the Institute of Electrical and Electronics Engineers are included in the MARC record to help users discover this journal.

37. Subjects of this journal are represented in Field 050 by a Library of Congress Classification (LCC) number of “TK5105.5” that represents general works on “Computer Networks,” in Field 082 by a Dewey Decimal Classification (DDC) number of “004.6” that represents “Interfacings and communications,” and in Field 650 by a Library of Congress subject heading that has “Computer networks” as the main topic, followed by a form subheading “Periodicals” encoded in subfield “v” (“|v”) to indicate the main topic is treated in a journal. These subject representations make it possible for users to conduct subject searches to discover this journal.

38. The Library of Congress records (**Appendix 1005B**) show that *IEEE Network* is a long-running journal, and the MARC record has made this journal discoverable in the library catalog since the serial record was created. Interested users would have been able to conduct subject searches by the LCC number, the DDC number, and the Library of Congress subject heading in this MARC record to discover this journal. They would also have been able to discover and access this publication by the title, variant titles, ISSN, the parent organization of the journal and the publisher.

39. The date stamp on the cover of vol. 8, no. 3 of *IEEE Network* that

contains *Mukherjee* (**Appendix 1005A**) shows the issue was received by Library of Congress on May 19, 1994. As discussed above, a customary library practice in processing journals is to use a serial check-in system to record the receipt of new issues. After a newly received issue goes through the serial check-in process, the issue is promptly processed and placed, usually within a week after serial check in, in the periodical reading room if the library has one, or on the shelf with earlier issues for easy user access. This means that after vol. 8, no. 3 that contains *Mukherjee* would have become publicly accessible by May 26, 1994, at Library of Congress.

D. Copyright Registration Record

40. **Appendix 1005C** is a true and correct copy of the copyright registration record for vol. 8, no. 3 of *IEEE Network*. I obtained this record from the public catalog of the U.S. Copyright Office by first accessing their website at <http://www.copyright.gov/>, selecting the “Learn More” link under “Search Copyright Records,” and following the link for “Official Public Catalog (1978 to present)” under “Online Records Collections” to perform a title search for the record. I personally located, identified and obtained this record. This is the type of record experts in my field would reasonably rely upon when forming their opinions.

41. The “Type of Work” field identifies the publication as “Serial.” The application title, title, serial publication year, imprint, publication history, and

copyright claimant match the information presented in the Library of Congress copy (**Appendix 1005A**). The “Issues Registered” field shows “v. 8, no. 3, May-Jun94.; Pub. 1994-05-10; Reg. 1994-09-23; TX0003882007” that means vol. 8, no. 3 was published on May 10, 1994, and registered with the US Copyright Office on September 23, 1994, and the registration number is TX0003882007.

E. Linda Hall Library Copy

42. To determine if *Mukherjee* was available in other libraries, I used the WorldCat record for this journal (<https://www.worldcat.org/title/14393593>) to identify Linda Hall Library as another holding library. After I verified the holdings information in the library’s online catalog, I requested a copy of *Mukherjee* from their Document Delivery Service. This library copy is presented as **Appendix 1005D** in this declaration.

43. **Appendix 1005D** is a true and correct copy of *Mukherjee* obtained from Linda Hall Library. Pages 1 to 16 (internal pages 26 to 41) are the full text of this article. I have closely compared this copy with the publisher copy (**EX1005**) and the Library of Congress copy (**Appendix 1005A**), and found the three copies to have the same content. The two library copies are identical in that they both show the tables and figures in color, while the publisher copy shows the illustrations in black and white.

44. Pages 17 to 19 are the issue cover, table of contents and copyright page

of vol. 8, no. 3 (May/June 1994) of *IEEE Network* that contains *Mukherjee*. These three pages are identical to the corresponding pages in the Library of Congress copy (**Appendix 1005A**). The only difference is that page 17 of the Linda Hall Library copy is the issue cover that carries a date stamp of “Linda Hall Library May 24 1994” and an address label indicating Linda Hall Library is the recipient of this issue.

F. Linda Hall Library Records

45. **Appendix 1005E** is a true and correct copy of the bibliographic and MARC records for *IEEE Network* that contains *Mukherjee*. I personally located, identified and obtained these records from the online catalog of the Linda Hall Library. These are the type of record experts in my field would reasonably rely upon when forming their opinion.

46. Pages 1 and 2 of **Appendix 1005E** are the bibliographic record that identifies this work as a journal and shows that it is available at “Linda Hall Library Closed Stacks – Serials (IEEE network.) and other location” and the library holds volumes 1 (1987) to 10 (1996). The record also shows the title, other titles, creators, identifiers, publisher, creation date, publication frequency, general notes and citation/references note.

47. Page 3 is the MARC record for this journal. Fields 008 and 040 show that this MARC record is a copy cataloging record based on the original MARC record created by “NSD” (the National Serial Data Program of the Library of

Congress), modified by “DLC” (the OCLC library code for Library of Congress), and then modified by “LHL,” which is the OCLC library symbol for Linda Hall Library. Because the Linda Hall Library MARC record (**Appendix 1005E**) and the Library of Congress MARC record (**Appendix 1005B**) are based on the same NSD record for copy cataloging, these two library records have a lot in common, including the ISSN in Field 022, the title in Field 245, variant titles in Fields 246, publisher information in Field 260, beginning issue and date in Field 362, source of title in Field 500, and the inclusion of the IEEE Communications Society and the Institute of Electrical and Electronics Engineers in Fields 710 as additional data elements for users to discover this journal. The differences are that the Linda Hall Library MARC record does not use a classification number, because they shelf journals by title; the publication frequency note reflects an older frequency; the parent organization is encoded in Field 550; and the record includes an index note in Field 510.

48. This MARC record of the Linda Hall Library (**Appendix 1005E**) shows that *IEEE Network* is a long-running journal that has been discoverable at the library since the record was created. Interested users would have been able to use the ISSN, title, variant titles, the parent organization of the journal and the publisher to discover this journal, and they would have been able to use the “Index to IEEE publications” to discover articles published in this journal.

49. The date stamp of vol. 8, no. 3 that contains *Mukherjee* (**Appendix**

1005D) shows that it was received by Linda Hall Library on May 24, 1994. As discussed earlier, a customary library practice in processing serials is to use a serial check-in system to record the receipt of issues. After a newly received issue has gone through the check-in system, it is processed quickly to make it publicly accessible, usually within a week after serial check-in is completed. It is therefore my opinion that the copy of vol. 8, no. 3 of *IEEE Network* (and *Mukherjee* contained therein) would have become publicly accessible no later than May 31, 1994, at this library.

G. Usage Record

50. Actual usage of a publication is reflected by the papers that make reference to it. The citation history on Google Scholar shows that *Mukherjee* has been cited 1,617 times, and seven citations were published in 1995. **Appendix 1005F** is a list I compiled from Google Scholar to demonstrate early usage. The earliest citation appeared in June 1995.

H. Summary of My Opinion on *Mukherjee*

51. The copyright registration record for *IEEE Network* (**Appendix 1005C**) shows that vol. 8, no. 3 that contains *Mukherjee* was published on May 10, 1994. The Library of Congress date stamp shows that their copy of *Mukherjee* (**Appendix 1005A**) was received on May 19, 1994, and the Linda Hall Library date stamp (**Appendix 1005D**) shows that their copy was received on May 24, 1994. It is therefore my opinion that vol. 8, no. 3 of *IEEE Network* that contains *Mukherjee*

would have been received by libraries in May 1994, and become publicly accessible no later than May 31, 1994. Citation history shows that the earliest citation was published in June 1995.

III. Conclusion

52. In signing this declaration, I recognize that the declaration will be filed as evidence in a Petition for *Inter Partes* Review before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to deposition in the case. If deposition is required of me, I will appear at a reasonable time and place to be agreed upon.

53. I hereby declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: 3-23-2023

Executed: Ingrid Hsieh-Yee
Ingrid Hsieh-Yee, Ph.D.