

Appendix F – Combination References

Fortinet, Inc. ("Fortinet") provides this chart subject to all reservations, objections, statements, and disclaimers set forth herein and in Fortinet's Preliminary Invalidity Contentions Cover Pleading, as well as any amendment, supplement, or modification thereof, which are incorporated herein by reference in their entirety.

On information and belief, and subject to further investigation and discovery, the following references are prior art to the '936 patent:

- U.S. Patent No. 6,591,377 to Evoy et al. ("Evoy"), filed November 24, 1999
- U.S. Patent Pub. No. 2005/0005171 to Oliphant et al. ("Oliphant"), which claims priority to July 1, 2003
- U.S. Patent App. No. 60/484,085 ("Oliphant Provisional"), filed July 1, 2003
- U.S. Patent No. 7,788,699 to Largman et al. ("Largman"), which claims priority to March 6, 2002
- U.S. Patent No. 9,503,470 to Gertner et al. ("Gertner"), which claims priority to December 24, 2002

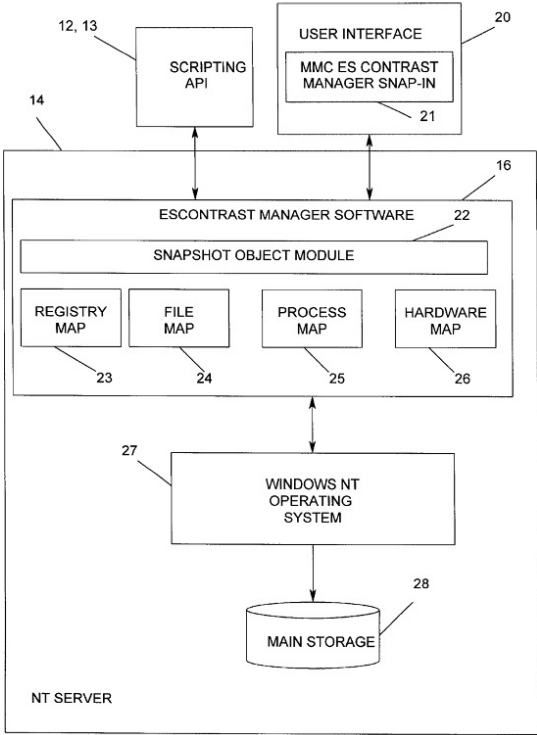
As illustrated in the chart below, these references disclose or render obvious certain limitations or claims of the '936 patent and can be combined with each other or other charted references for the reasons set forth in the Cover Pleading. These Preliminary Invalidity Contentions are not an admission by Fortinet that the accused products, including any current or past versions of the accused products, are covered by, or infringe the asserted claims, but are based instead on the recognition that, if the claims are interpreted to be broad enough to encompass the accused products, the claims must also be construed to have that same scope when considering whether they are invalid.

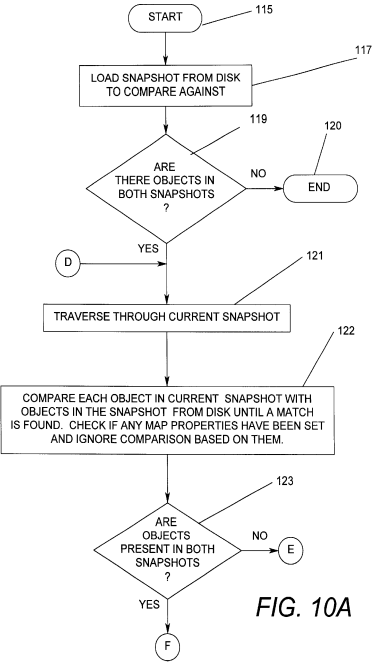
The following chart is partially based on, but is not limited by, the claim constructions implicit in Plaintiff's Infringement Contentions, to the extent that such constructions are apparent from the Infringement Contentions. Fortinet notes that in many instances, Plaintiff's Infringement Contentions fail to provide adequate notice of Plaintiff's construction of the asserted claims and fail to comply with the Court's scheduling order and other applicable rules. Fortinet does not accept the assumptions concerning the scope and meaning implicit in Plaintiff's Infringement Contentions, to the extent those assumptions are discernible, and reserves the right to challenge Plaintiff's proposed (or implied) constructions. Fortinet also reserves the right to revise and supplement these charts if and when Plaintiff is permitted to provide revised Infringement Contentions or otherwise make its positions known. To the extent that these Preliminary Invalidity Contentions rely on or otherwise embody particular constructions of terms or phrases in the asserted claims, Fortinet does not necessarily advocate any such construction as proper constructions of those terms or phrase. Fortinet also reserves

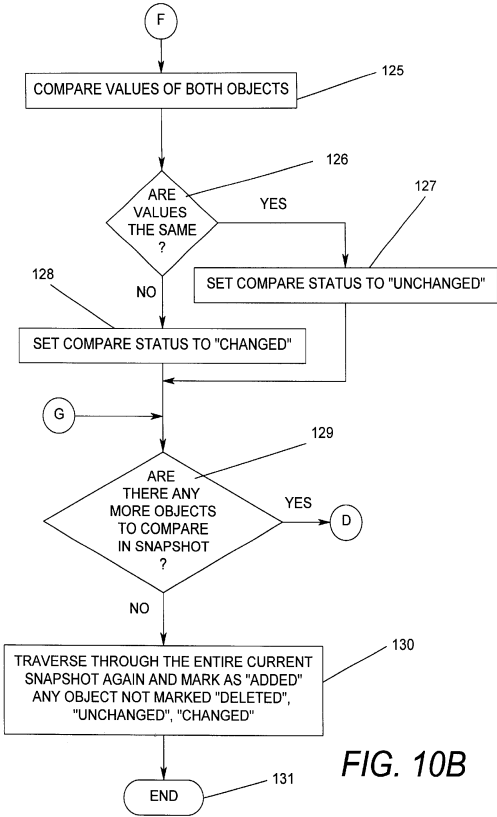
the right to revise and supplement these charts after the Court construes the claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Claim	Exemplary Citation from [REFERENCE]
[1PRE] A method of detecting abnormal system states in computers, comprising:	
[1A] receiving snapshots from a plurality of computers within a population of computers, wherein individual snapshots include data indicating a state of a respective computer;	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards, printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information." Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided." Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond, Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at Fig. 2 and corresponding text:  <pre> graph TD subgraph NT_SERVER [NT SERVER] subgraph UI [USER INTERFACE 20] MMC[MMC ES CONTRAST MANAGER SNAP-IN 21] end subgraph API [SCRIPTING API 12, 13] end subgraph EScontrast [ESCONTRAST MANAGER SOFTWARE 22] subgraph SOM [SNAPSHOT OBJECT MODULE 22] REG[REGISTRY MAP 23] FILE[FILE MAP 24] PROC[PROCESS MAP 25] HARD[HARDWARE MAP 26] end end subgraph OS [WINDOWS NT OPERATING SYSTEM 27] end subgraph STORAGE [MAIN STORAGE 28] end end UI <--> EScontrast API <--> EScontrast EScontrast <--> OS OS --> STORAGE </pre> FIG. 2
[1B] automatically generating an adaptive reference model comprising a rule set customized to characteristics of the population of computers, the rule set being developed by identifying patterns among the snapshots from	

Claim	Exemplary Citation from [REFERENCE]
the plurality of computers such that the adaptive reference model is indicative of normal states in the computers within the population; and	
[1C] comparing a snapshot from at least one of the computers to the adaptive reference model to determine whether an anomaly is present in the state of the at least one of the computers.	Evoy at Fig. 10A and corresponding text:  <pre> graph TD 115([START]) --> 117[LOAD SNAPSHOT FROM DISK TO COMPARE AGAINST] 117 --> 119{ARE THERE OBJECTS IN BOTH SNAPSHOTS?} 119 -- NO --> 120([END]) 119 -- YES --> 121[TRAVERSE THROUGH CURRENT SNAPSHOT] 121 --> 122[COMPARE EACH OBJECT IN CURRENT SNAPSHOT WITH OBJECTS IN THE SNAPSHOT FROM DISK UNTIL A MATCH IS FOUND. CHECK IF ANY MAP PROPERTIES HAVE BEEN SET AND IGNORE COMPARISON BASED ON THEM.] 122 --> 123{ARE OBJECTS PRESENT IN BOTH SNAPSHOTS?} 123 -- YES --> F((F)) 123 -- NO --> E((E)) </pre> FIG. 10A

Claim	Exemplary Citation from [REFERENCE]
	Evoy at Fig. 10B and corresponding text:  <pre> graph TD F((F)) --> 125[COMPARE VALUES OF BOTH OBJECTS] 125 --> 126{ARE VALUES THE SAME?} 126 -- YES --> 127[SET COMPARE STATUS TO "UNCHANGED"] 126 -- NO --> 128[SET COMPARE STATUS TO "CHANGED"] 127 --> 129{ARE THERE ANY MORE OBJECTS TO COMPARE IN SNAPSHOT?} 128 --> 129 129 -- YES --> D((D)) 129 -- NO --> 130[TRAVERSE THROUGH THE ENTIRE CURRENT SNAPSHOT AGAIN AND MARK AS "ADDED" ANY OBJECT NOT MARKED "DELETED", "UNCHANGED", "CHANGED"] 130 --> 131((END)) </pre> FIG. 10B Evoy at 7:28: "A Compare method 56 of the snapshot object 29 can be used to compare snapshots of System States taken at different times. A Compare method is preceded by a Load method (within bracket 72), which is used to load a previously taken snapshot from the disk and load it into memory. Subsequently, Compare calls the compare methods in each of the individual map objects to compare the collection of the currently held System State objects in the map objects with the System State objects of the snapshot loaded into the memory. The result of this compare operation is that any changes, additions

Claim	Exemplary Citation from [REFERENCE]
	or deletions of objects and their values between the two snapshots is marked out in the snapshot currently held by the map objects. Thus, after the Compare operation, the collection of System State objects held by the map objects reflect the changes, if any, between the two snapshots. These changes can be restored, so that the current snapshot is returned to its original state by using a Restore method (within the bracket 72). Also, a Save_As method 72 can be used to save any snapshot into the disk." Evoy at 9:31: "On the other hand, if there are objects in both snapshots, then each object in the snapshot from the disk is compared with objects in the current snapshot. Also, a check is made as to whether or not any map properties have been set that would affect the comparison of the file and process data items as described for blocks 66 and 67 of FIG. 5 (block 122). After this, another inquiry is made as to whether or not objects are present in both snapshots (diamond 123). If the answer to this inquiry is yes, then the process illustration continues in FIG. 10B at a connector A. On the other hand, if the answer to this inquiry is no, then the process illustration continues in FIG. 10C at a connector B." Evoy at 9:42: "Referring now to FIG. 10B at the connector A, the values of both objects are compared (block 125). Next, an inquiry is made as to whether or not values of the objects are the same (diamond 126). If the answer to this inquiry is yes, then compare status of the object in the current snapshot is set to "Unchanged" (block 127). On the other hand, if the answer is no, then compare status of the object in the current snapshot is set to "Changed" (block 128)."
[2PRE] The method of claim 1, further comprising:	See claim 1.
[2A] comparing the anomaly to a recognition filter to diagnose a trouble condition on the at least one of the computers; and	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of

Claim	Exemplary Citation from [REFERENCE]
	open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

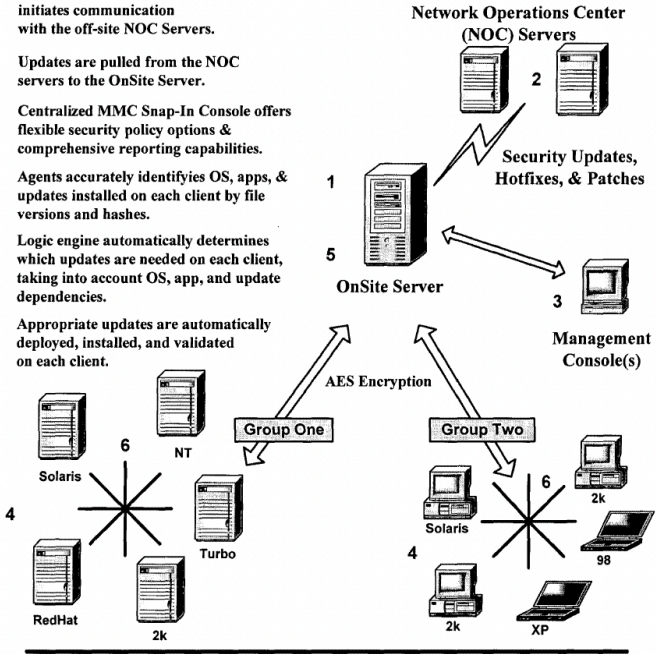
Claim	Exemplary Citation from [REFERENCE]
	• RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed • RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client • RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group • RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client • RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group • InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client • InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group • InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client • InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. • GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

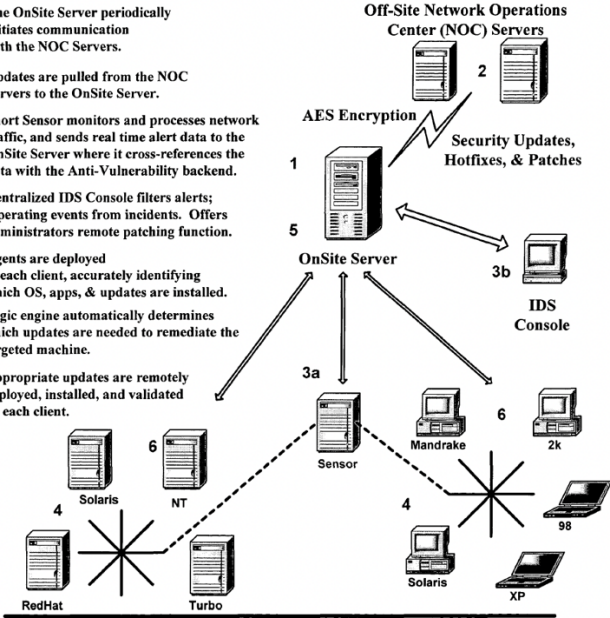
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

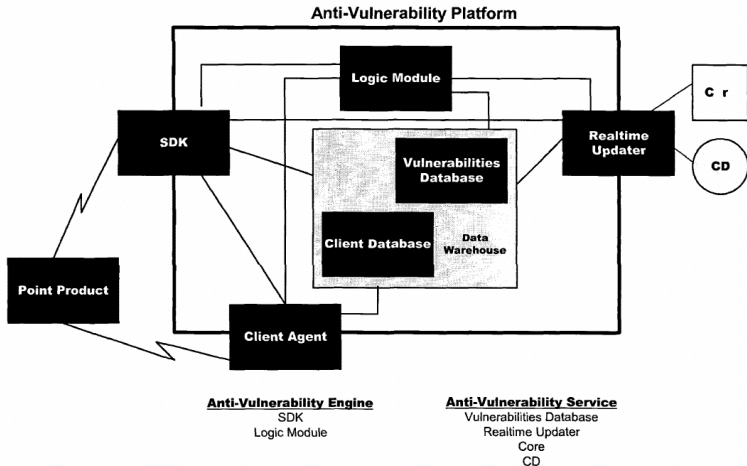
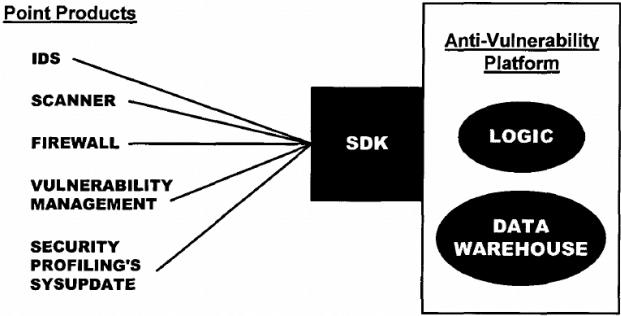
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 35: Solution SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below. Functional Description 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. Unique Features: • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption'. The OnSite Server is also connected to an 'IPS Console'. The OnSite Server (1) is connected to an 'Inline Sensor' (labeled 3a and 3b). The Inline Sensor is connected to a 'Perimeter Firewall'. The Perimeter Firewall is connected to a 'Hub'. The Hub is connected to two groups of client machines: one group (labeled 4) includes Solaris, NT, RedHat, and Turbo; the other group (labeled 6) includes Mandrake, 2k, 98, and Solaris. The diagram is numbered 1 through 6, corresponding to the steps in the list below. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  The diagram illustrates the SysUpdate™ network configuration. At the top, 'Network Operations Center (NOC) Servers' (labeled 2) send 'Security Updates, Hotfixes, & Patches' to an 'OnSite Server' (labeled 5). The OnSite Server communicates with a 'Management Console(s)' (labeled 3). The OnSite Server then distributes updates to two client groups, 'Group One' and 'Group Two', using 'AES Encryption'. Group One includes clients running Solaris, NT, Turbo, RedHat, and 2k. Group Two includes clients running Solaris, 2k, XP, and 98. A central star symbol in each group indicates the distribution point. The diagram is accompanied by a list of six numbered steps: 1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 5) via 'AES Encryption' (labeled 1). The 'OnSite Server' is also connected to an 'IDS Console' (labeled 3b). A 'Sensor' (labeled 3a) is connected to the 'OnSite Server' and a network of client machines. The client machines include Solaris, NT, RedHat, Turbo, Mandrake, 2k, 98, and XP. The diagram is accompanied by a list of six numbered steps: 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[2B] in the event of a trouble condition, generating an automated response to the trouble condition.	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

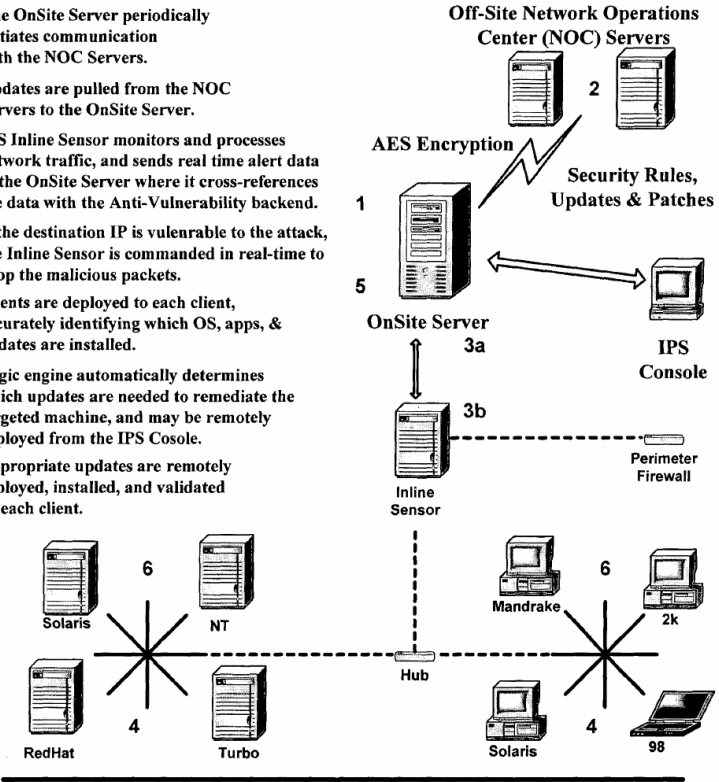
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

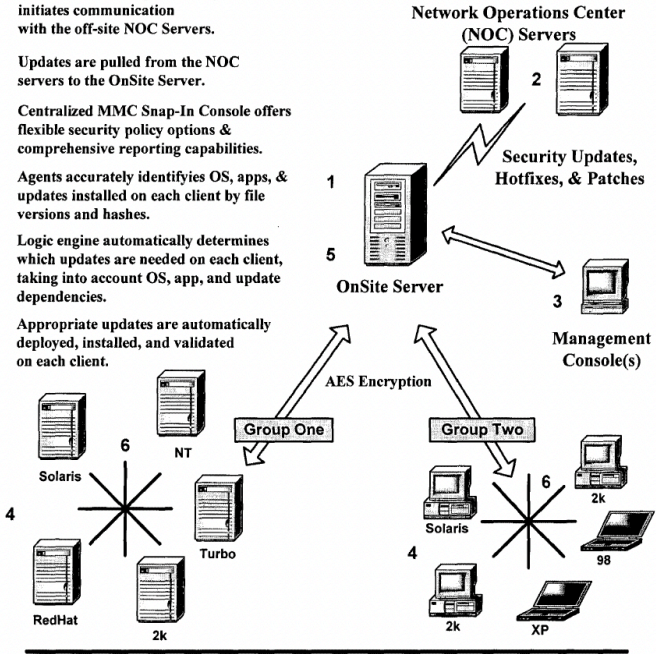
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

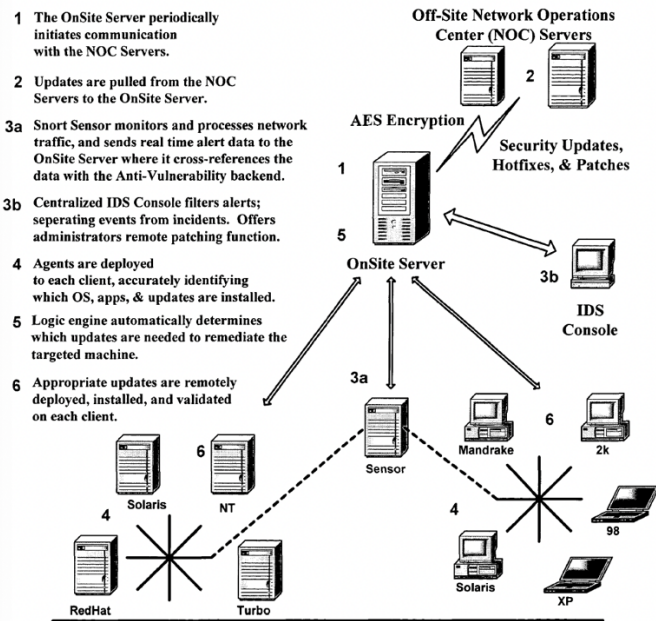
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

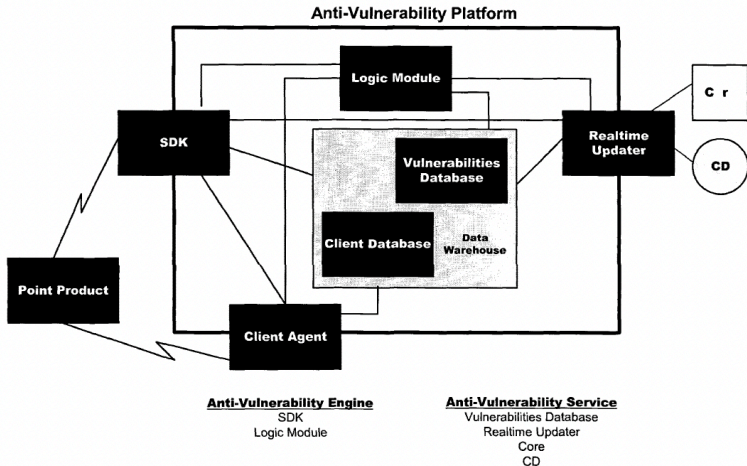
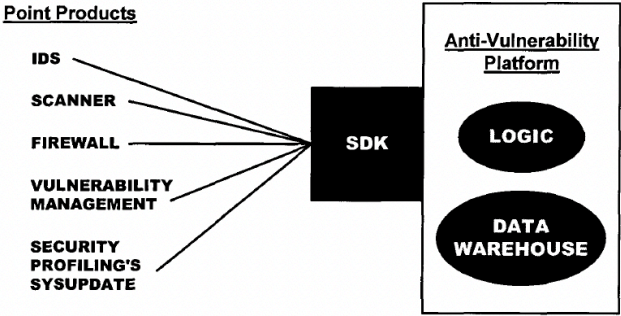
Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram  The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected via 'AES Encryption' to an 'OnSite Server' (labeled 1 and 5). The 'OnSite Server' is also connected to an 'IPS Console'. Below the 'OnSite Server' is an 'Inline Sensor' (labeled 3b), which is connected to a 'Perimeter Firewall'. The 'Perimeter Firewall' is connected to a 'Hub'. The 'Hub' is connected to two groups of client computers. The left group includes 'Solaris', 'NT', 'RedHat', and 'Turbo' (labeled 4). The right group includes 'Mandrake', '2k', 'Solaris', and '98' (labeled 4). The 'OnSite Server' is also connected to the 'Hub' via a dashed line. The diagram is divided into two main sections by a horizontal line. The top section shows the communication between the 'OnSite Server' and the 'NOC Servers'. The bottom section shows the network topology, including the 'Perimeter Firewall', 'Hub', and client computers. The numbered steps 1 through 6 are listed on the left side of the diagram, describing the process of updating and monitoring the network. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 Solaris NT Turbo RedHat 2k 4 Solaris 2k XP 98

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 5) via 'AES Encryption' (labeled 1). The 'OnSite Server' is also connected to an 'IDS Console' (labeled 3b). A 'Sensor' (labeled 3a) is connected to the 'OnSite Server' and a central network hub. The hub is connected to various client machines: 'RedHat', 'Solaris', 'NT', 'Turbo', 'Mandrake', '2k', '98', and 'XP'. The diagram is accompanied by a list of six numbered steps: 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[3PRE] The method of claim 2, wherein the automated response is a generic response not specific to a particular asset of the at least one computer, and wherein the method further comprises:	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

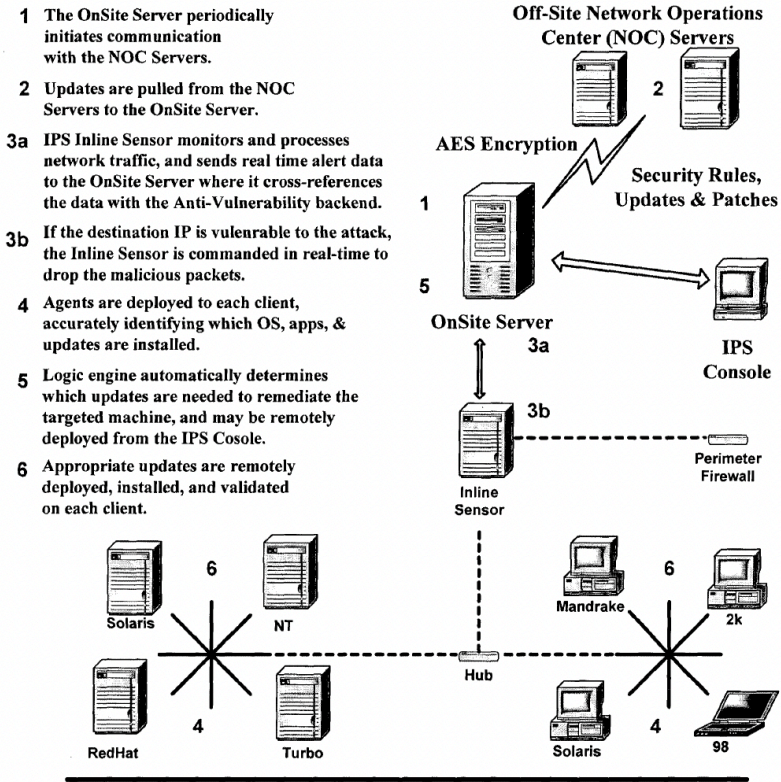
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

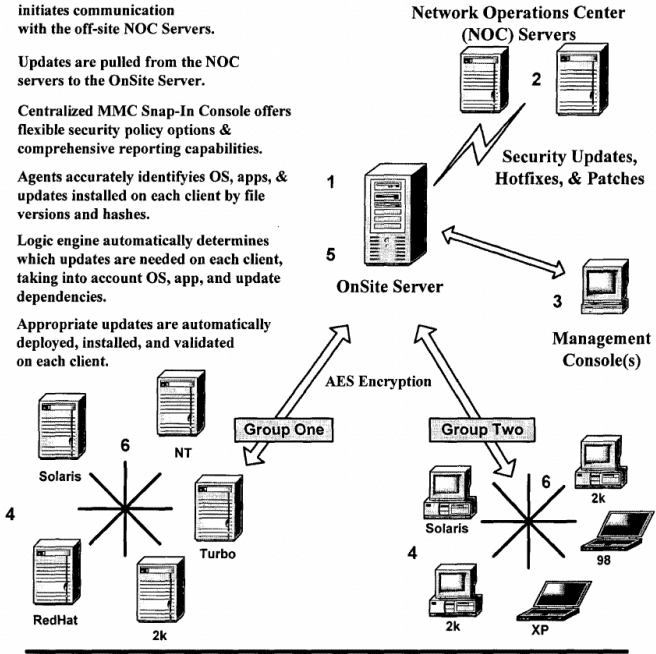
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

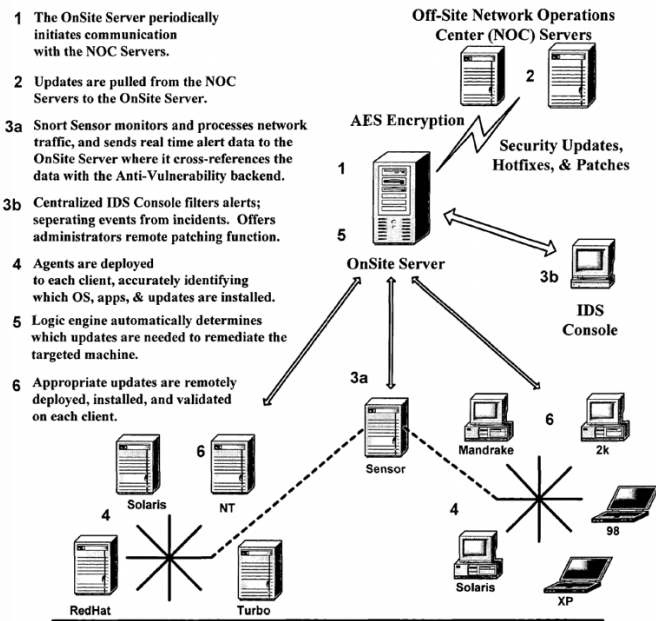
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: atform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

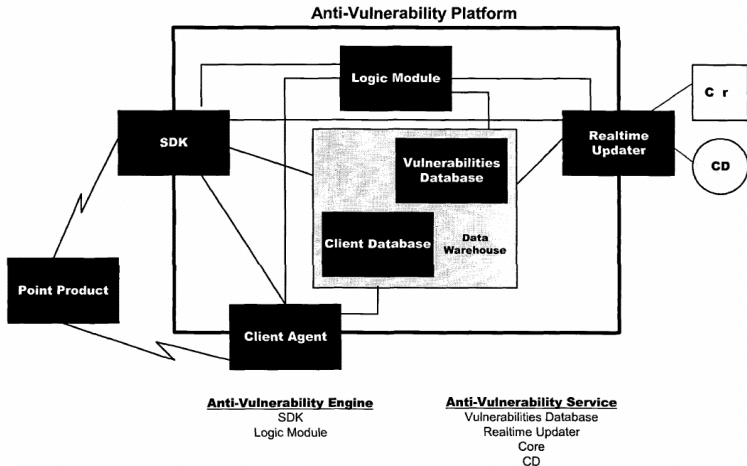
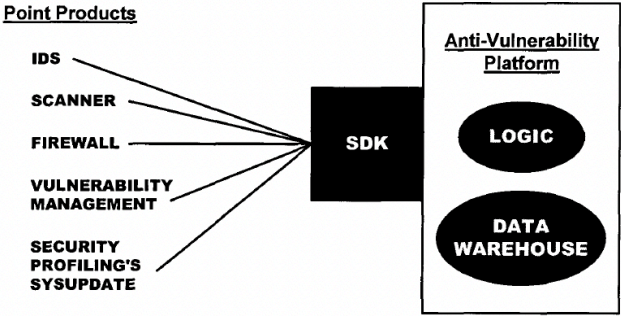
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 35: Solution SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below. Functional Description 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. Unique Features: • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram  The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected via 'AES Encryption' to an 'OnSite Server' (labeled 1 and 5). The OnSite Server is also connected to an 'IPS Console'. Below the OnSite Server is an 'Inline Sensor' (labeled 3a and 3b), which is connected to a 'Perimeter Firewall'. The network is divided into two segments by the firewall, each connected to a 'Hub'. The left segment contains four servers: Solaris, NT, RedHat, and Turbo, with a star symbol and the number 4. The right segment contains four desktop computers: Mandrake, 2k, Solaris, and 98, with a star symbol and the number 4. A large number 6 is placed near the hubs on both sides. The numbered steps on the left describe the system's operation: 1. OnSite Server communicates with NOC Servers; 2. Updates are pulled from NOC Servers; 3a. IPS Inline Sensor monitors traffic and sends alert data; 3b. Inline Sensor is commanded to drop malicious packets; 4. Agents are deployed to clients; 5. Logic engine determines updates; 6. Updates are remotely deployed and validated.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 Solaris NT Turbo RedHat 2k 4 Solaris 2k XP 98

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 5) via 'AES Encryption' (labeled 1). The 'OnSite Server' is also connected to an 'IDS Console' (labeled 3b). A 'Sensor' (labeled 3a) is connected to the 'OnSite Server' and a network of client machines. The client machines include Solaris, NT, RedHat, Turbo, Mandrake, 2k, 98, and XP. The diagram is accompanied by a list of six numbered steps: 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[3A] sending the generic response and a set of anomalies found in the snapshot to the at least one computer, the set of anomalies indicating assets of the at least one computer whose states are anomalous; and	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

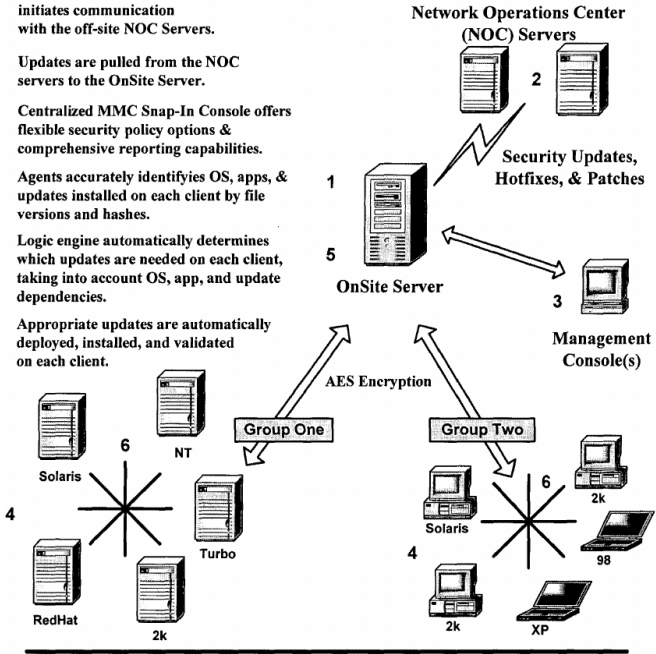
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

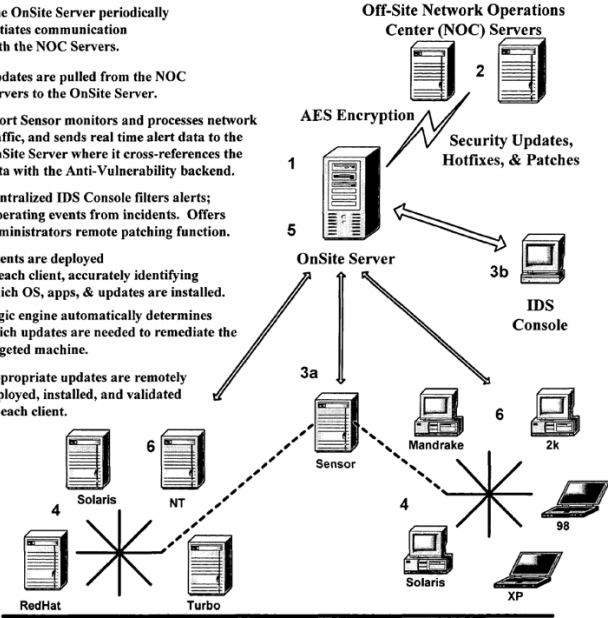
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: atform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

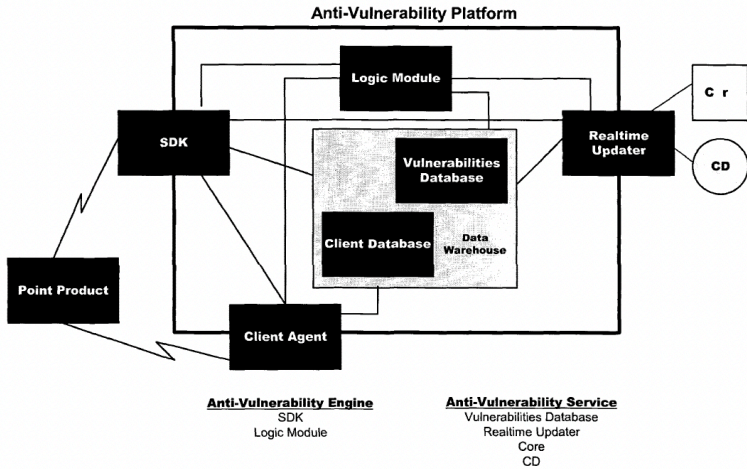
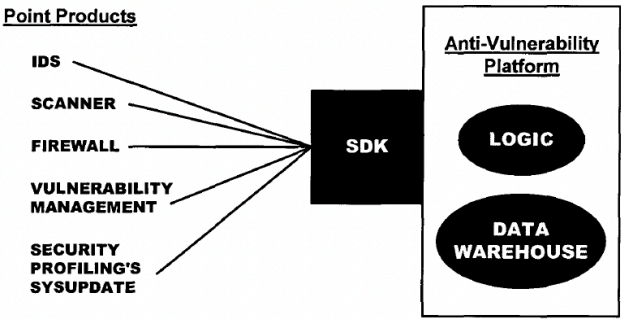
Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> <div data-bbox="1079 365 1308 690"> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption </div>

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption'. The OnSite Server is also connected to an 'IPS Console'. The OnSite Server (1) sends 'Security Rules, Updates & Patches' to the IPS Console. The OnSite Server (5) is connected to an 'Inline Sensor' (labeled 3a and 3b). The Inline Sensor is connected to a 'Perimeter Firewall'. The Perimeter Firewall is connected to a 'Hub'. The Hub is connected to two groups of client machines. The left group includes 'Solaris', 'NT', 'RedHat', and 'Turbo' (labeled 4 and 6). The right group includes 'Mandrake', '2k', 'Solaris', and '98' (labeled 4 and 6). The diagram is divided into two main sections by a horizontal line, with the top section showing the server and console components and the bottom section showing the client machines and network topology. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 Solaris NT Turbo RedHat 2k 4 Solaris 2k 98 2k XP

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates the architecture of an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 5) via 'AES Encryption' (labeled 1). The OnSite Server is also connected to an 'IDS Console' (labeled 3b). The OnSite Server is connected to a 'Sensor' (labeled 3a), which in turn is connected to a central hub. This hub is connected to various client machines, including Solaris, NT, RedHat, Turbo, Mandrake, 2k, 98, and XP. The diagram is accompanied by a list of six numbered steps describing the process. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  For more detail, see the Platform Component Diagram on page 8.
[3B] applying the generic response to the anomalous	Oliphant at ¶ 22-23:

Claim	Exemplary Citation from [REFERENCE]
assets to correct the trouble condition.	"In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)."

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities. Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

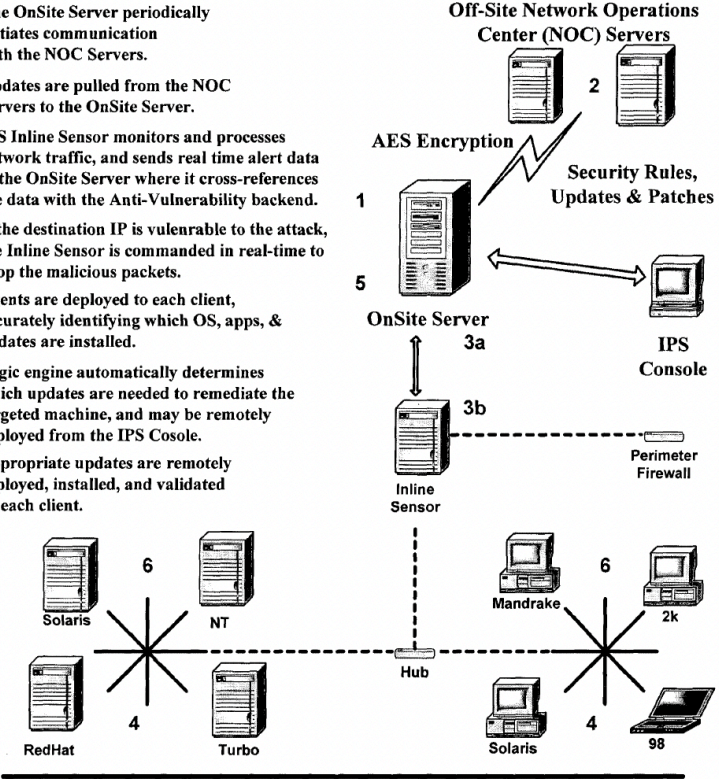
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

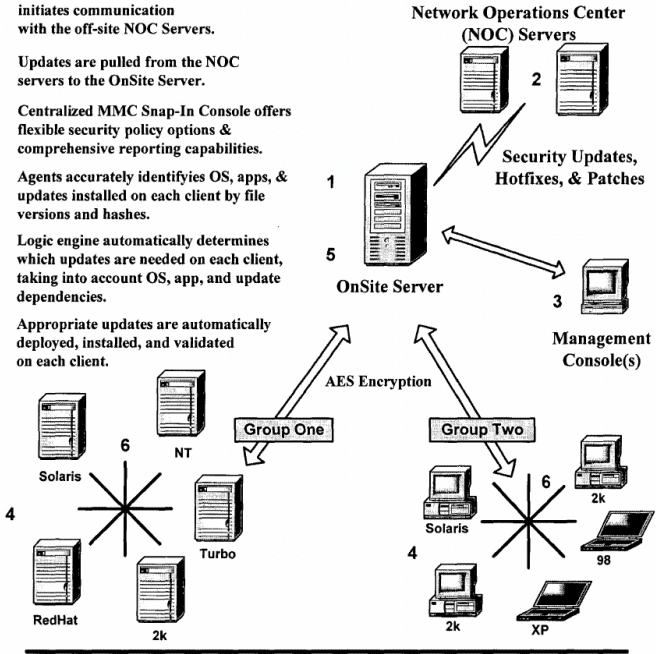
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

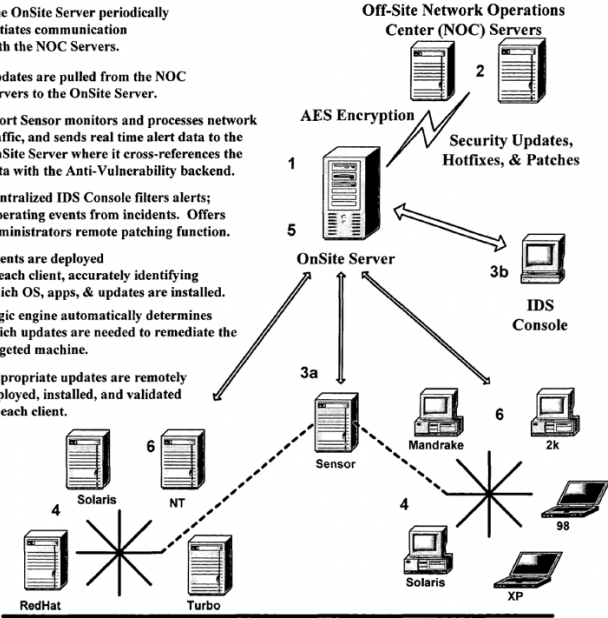
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: atform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

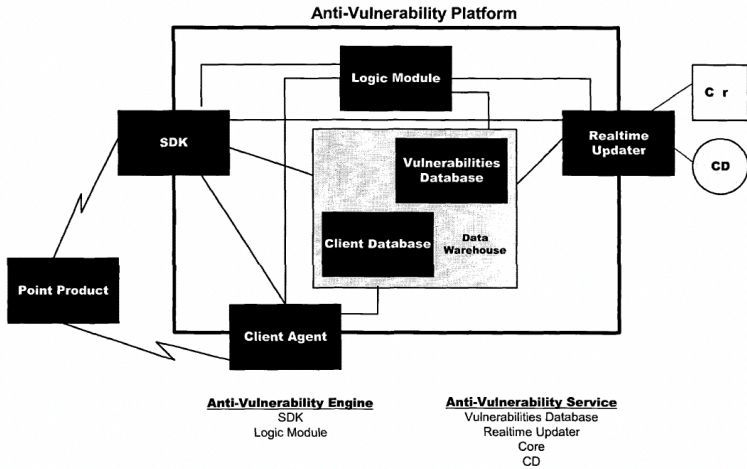
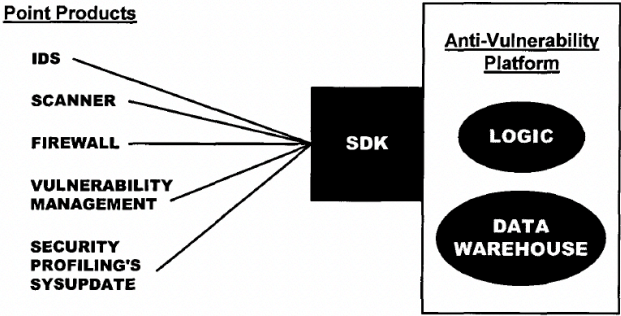
Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> <div data-bbox="1079 365 1308 690"> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption </div>

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram  The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption'. The OnSite Server is also connected to an 'IPS Console'. The OnSite Server (1) is connected to an 'Inline Sensor' (labeled 3a and 3b). The Inline Sensor is connected to a 'Perimeter Firewall'. The Perimeter Firewall is connected to a 'Hub'. The Hub is connected to two groups of client machines. The left group includes 'Solaris', 'NT', 'RedHat', and 'Turbo' (labeled 4 and 6). The right group includes 'Mandrake', '2k', 'Solaris', and '98' (labeled 4 and 6). The diagram is divided into two main sections by a horizontal line, with the top section showing the server and console components and the bottom section showing the client machines and network topology. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 Solaris NT Turbo RedHat 2k 4 Solaris 2k XP 98

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1) via 'AES Encryption'. The 'OnSite Server' is also connected to an 'IDS Console' (labeled 3b). A 'Sensor' (labeled 3a) is connected to the 'OnSite Server' and a network of client machines. The client machines include Solaris, NT, RedHat, Turbo, Mandrake, 2k, 98, and XP. The diagram is accompanied by a list of six numbered steps describing the system's operation. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[4] The method of claim 3, wherein the generic response includes at least one of: installing a missing software component, removing an undesirable software component, and restoring an incorrect registry setting.	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

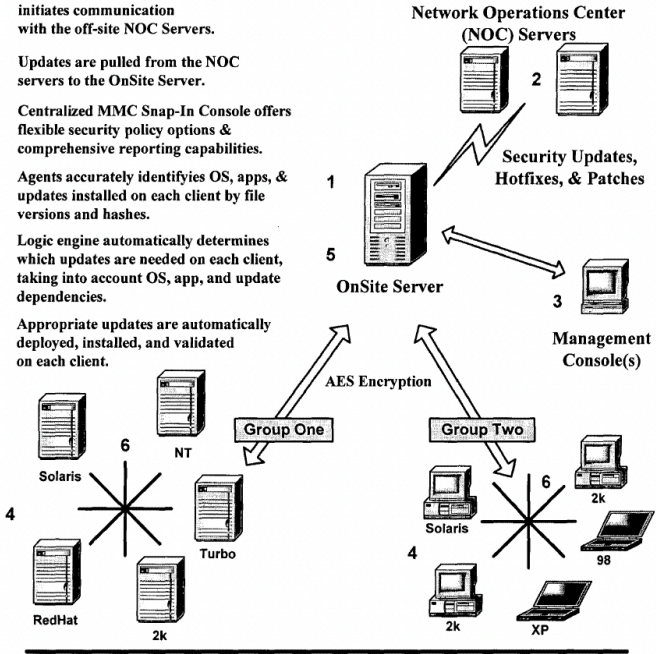
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> <div data-bbox="1079 365 1308 690"> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption </div>

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

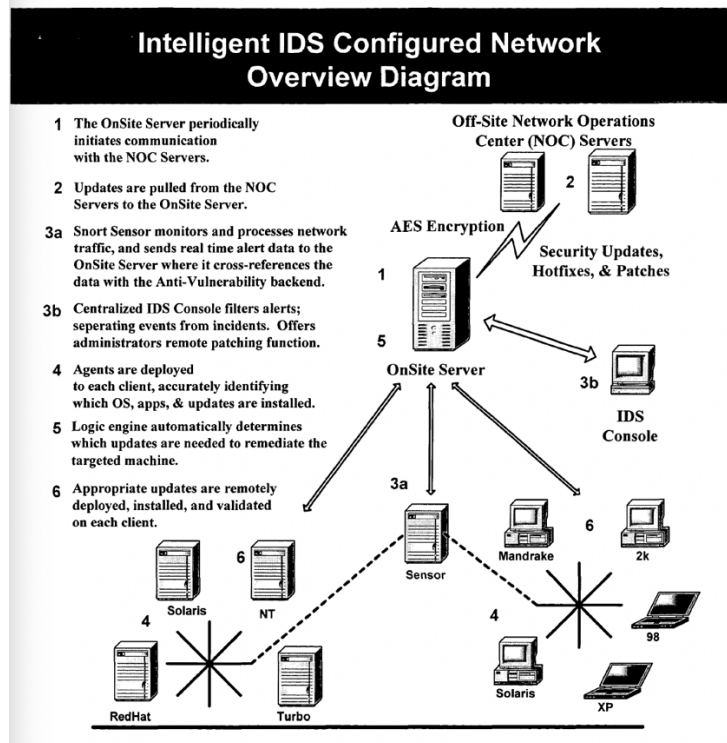
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption' and 'Security Rules, Updates & Patches'. The 'OnSite Server' is also connected to an 'IPS Console'. Below the 'OnSite Server' is an 'Inline Sensor' (labeled 3b), which is connected to a 'Perimeter Firewall'. The 'Perimeter Firewall' is connected to a 'Hub'. The 'Hub' is connected to two groups of client machines. The left group includes 'Solaris', 'NT', 'RedHat', and 'Turbo' (labeled 4). The right group includes 'Mandrake', '2k', 'Solaris', and '98' (labeled 4). The 'OnSite Server' is also connected to the 'Hub' via a dashed line. The 'OnSite Server' is labeled with '1' and '5', and the 'Inline Sensor' is labeled with '3a' and '3b'. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

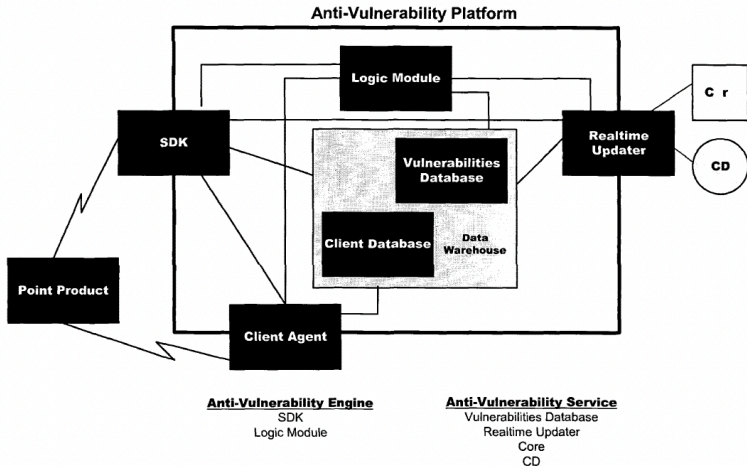
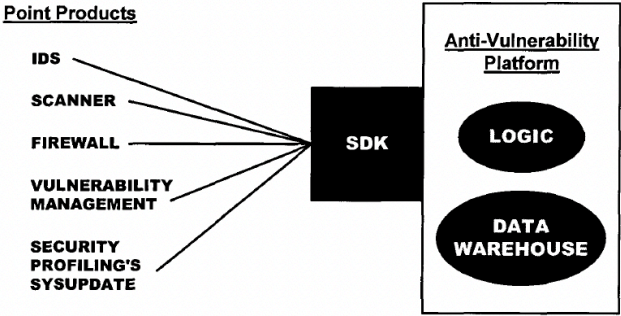
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers Security Updates, Hotfixes, & Patches OnSite Server Management Console(s) AES Encryption Group One Group Two 4 4 6 6 Solaris NT Turbo RedHat 2k Solaris 2k 2k XP 98

Claim

Exemplary Citation from [REFERENCE]

Oliphant Provisional at 34:



Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[5] The method of claim 2, wherein the recognition filter comprises a particular pattern of anomalies that indicates the presence of a particular root cause condition or a generic class of conditions.	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

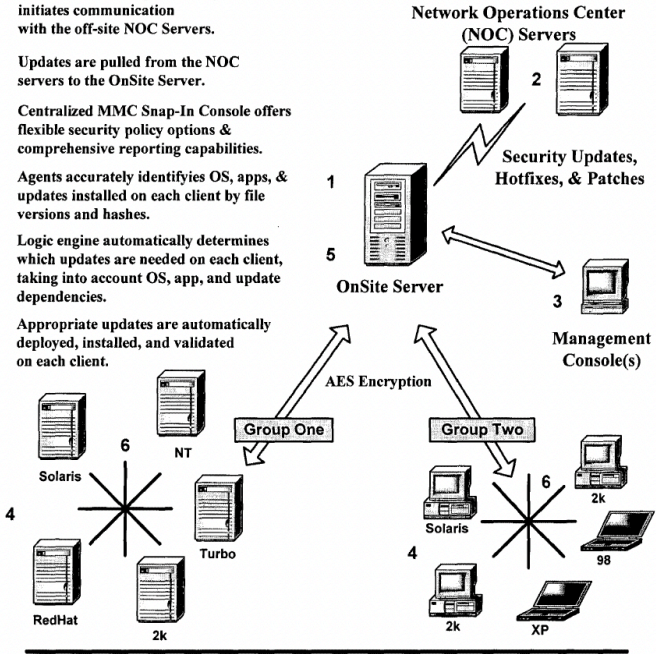
Claim	Exemplary Citation from [REFERENCE]
	RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

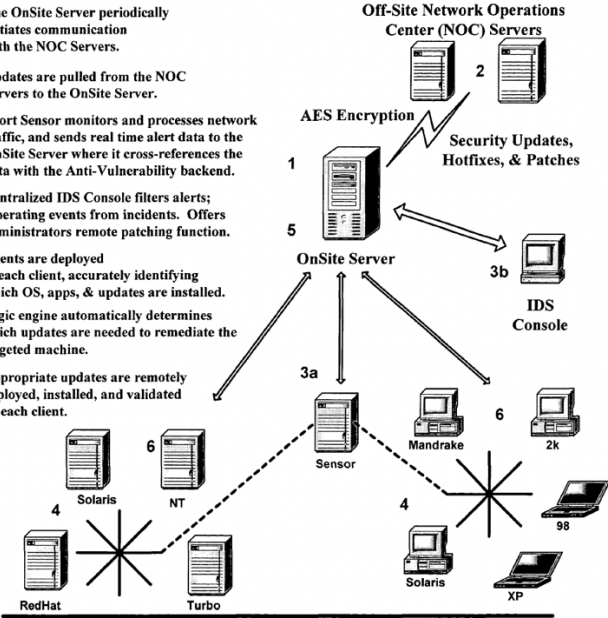
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

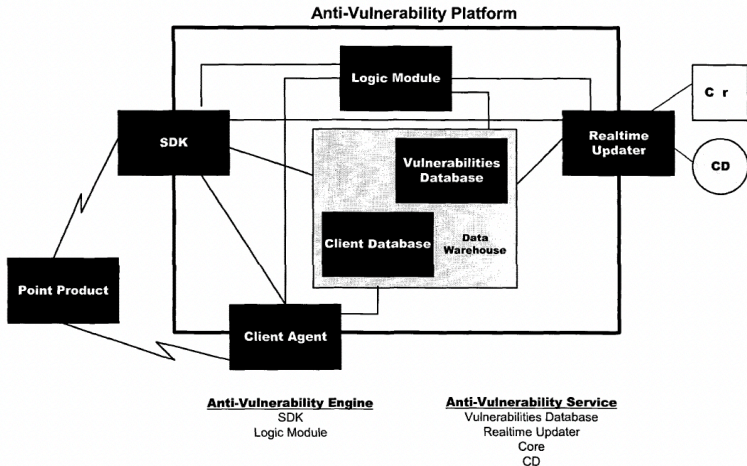
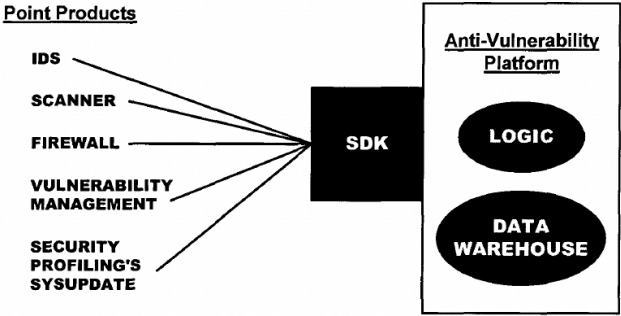
Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> <div data-bbox="1083 367 1308 690"> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption </div>

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client. Off-Site Network Operations Center (NOC) Servers 2 AES Encryption 1 Security Rules, Updates & Patches OnSite Server 3a IPS Console 3b Perimeter Firewall Inline Sensor Hub Solaris NT RedHat Turbo Mandrake 2k Solaris 98

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  The diagram illustrates the SysUpdate™ network configuration. At the top, 'Network Operations Center (NOC) Servers' (labeled 2) send 'Security Updates, Hotfixes, & Patches' to the 'OnSite Server' (labeled 5). The 'OnSite Server' is connected to a 'Management Console(s)' (labeled 3). The 'OnSite Server' also sends updates to two client groups, 'Group One' and 'Group Two', via 'AES Encryption'. 'Group One' includes clients running Solaris, NT, Turbo, RedHat, and 2k. 'Group Two' includes clients running Solaris, 2k, XP, and 98. The diagram is numbered 1 through 6, corresponding to the steps listed on the left. 1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client. Off-Site Network Operations Center (NOC) Servers AES Encryption Security Updates, Hotfixes, & Patches OnSite Server IDS Console Sensor RedHat, Solaris, NT, Turbo Mandrake, 2k, 98, XP

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[6PRE] The method of claim 1, further comprising:	See Claim 1.
[6A] comparing a plurality of anomalies associated with a particular snapshot with a recognition filter to diagnose a trouble condition; and	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies;

Claim	Exemplary Citation from [REFERENCE]
	and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

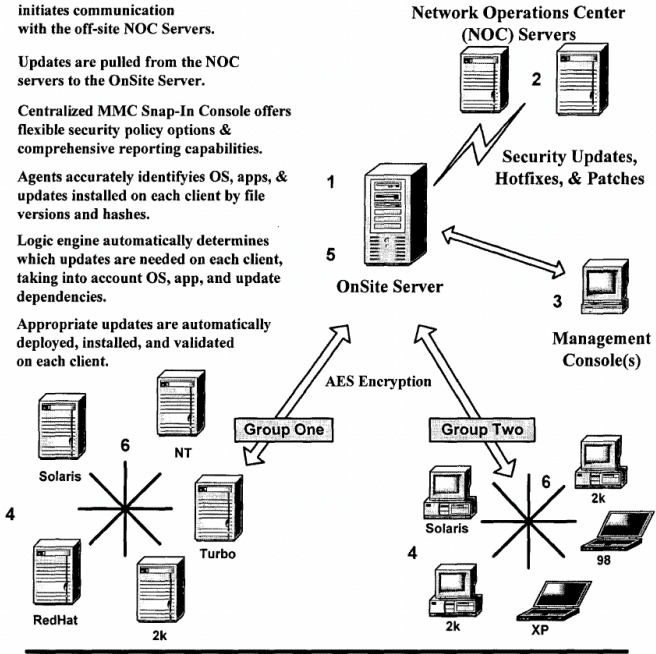
Claim	Exemplary Citation from [REFERENCE]
	• RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed • RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client • RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group • RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client • RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group • InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client • InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group • InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client • InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. • GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

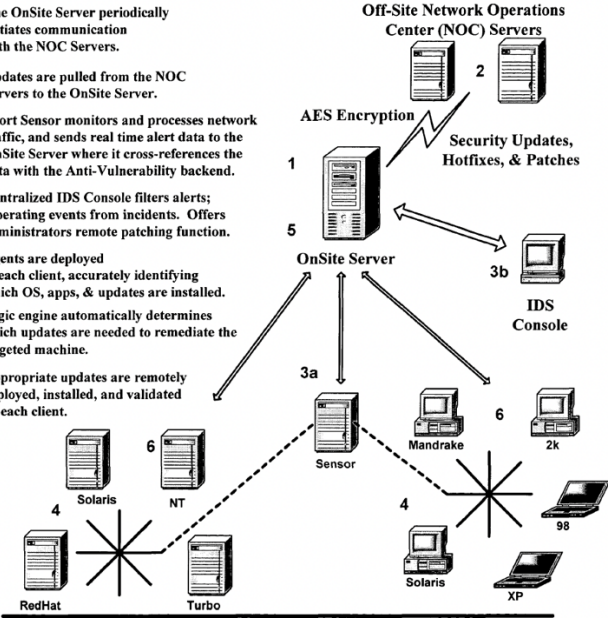
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

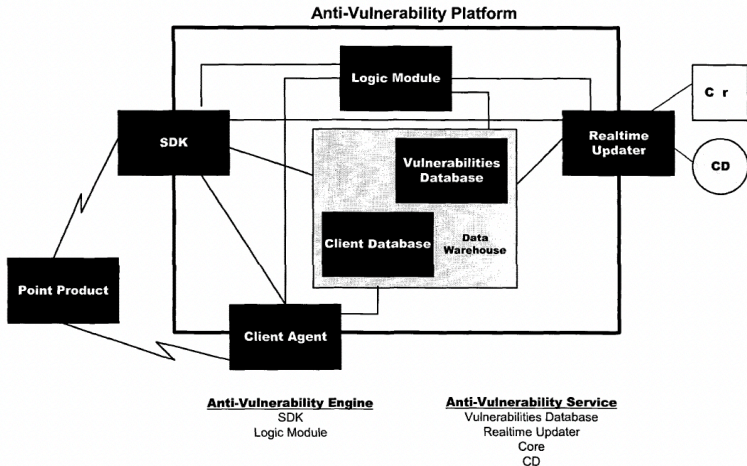
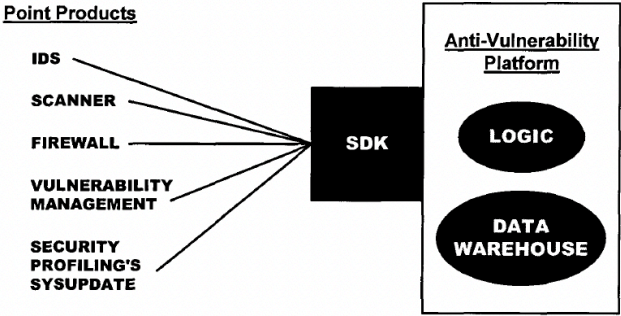
Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> <div data-bbox="1083 367 1308 690"> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption </div>

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption'. The OnSite Server is also connected to an 'IPS Console'. The OnSite Server (1) is connected to an 'Inline Sensor' (labeled 3a and 3b). The Inline Sensor is connected to a 'Perimeter Firewall'. The Perimeter Firewall is connected to a 'Hub'. The Hub is connected to two groups of client machines: one group (labeled 4) including Solaris, NT, RedHat, and Turbo; and another group (labeled 6) including Mandrake, 2k, 98, and Solaris. The diagram is numbered 1 through 6, corresponding to the steps in the list below. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 Solaris NT Turbo RedHat 2k 4 Solaris 2k 98 2k XP

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 34: Intelligent IDS Configured Network Overview Diagram  The diagram illustrates the architecture of an Intelligent IDS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1) via 'AES Encryption'. The OnSite Server is also connected to an 'IDS Console' (labeled 3b). A 'Sensor' (labeled 3a) is connected to the OnSite Server and a central network hub. The hub is connected to various client machines: RedHat, Solaris, NT, Turbo, Mandrake, 2k, 98, and XP. The diagram is accompanied by a list of six numbered steps describing the system's operation. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a Snort Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b Centralized IDS Console filters alerts; separating events from incidents. Offers administrators remote patching function. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[6B] diagnosing a trouble condition on the at least one of the computers in response to at least a subset of the plurality of anomalies matching information in the recognition filter.	Oliphant at ¶ 22-23: "In some embodiments, upon a determination by security server 135 that a connection attempt or other attack has occurred against a computer that is vulnerable (based on its current software, patch, policy, and configuration status), security server 135 selects one or more remediation techniques from database 146 that remediate the particular vulnerability. Based on a prioritization previously selected by an administrator or the system designer, the remediation technique(s) are applied (1) to the machine that was attacked, (2) to all devices subject to the same vulnerability (based on their real-time software, patch, policy, and configuration status), or (3) to all devices to which the selected remediation can be applied. In various embodiments, remediation techniques include the closing of open ports on the device; installation of a patch that is known to correct the vulnerability; changing the device's configuration; stopping, disabling, or removing services; setting or modifying policies; and the like. Furthermore, in various embodiments, events and actions are logged (preferably in a

Claim	Exemplary Citation from [REFERENCE]
	non-volatile medium) for later analysis and review by system administrators. In these embodiments, the log also stores information describing whether the target device was vulnerable to the attack." Oliphant at ¶ 11: "In particular, security server 135 includes processor 142, and memory 144 encoded with programming instructions executable by processor 142 to perform several important security-related functions. For example, security server 135 collects data from devices 131, 133, 137, and 139, including the software installed on those devices, their configuration and policy settings, and patches that have been installed. Security server 135 also obtains from vulnerability and remediation database 110 a regularly updated list of security vulnerabilities in software for a wide variety of operating systems, and even in the operating systems themselves. Security server 135 also downloads a regularly updated list of remediation techniques that can be applied to protect a device from damage due to those vulnerabilities. In a preferred embodiment, each vulnerability in remediation database 110 is identified by a vulnerability identifier, and the vulnerability identifier can be used to retrieve remediation information from database 110 (and from database 146, discussed below in relation to FIG. 2)." Oliphant Provisional at 9: What is Anti-Vulnerability™ Technology? Anti-Vulnerability technology is a platform and underlying back end that consists of a logic engine and vulnerability data warehouse. (Reference the diagram on page 8.) It will easily integrate with and enable network security products such as IDS, scanners, or firewalls to intelligently reference and share the same vulnerability data set, and independently provide complete vulnerability remediation (patching) functionalities such as that provided by the patch management and vulnerability remediation application, SysUpdate™. Thereby, the technology will improve system accuracy and efficiencies, minimize false positives and false negatives, and provide policy compliance and enforcement capabilities.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 9-10: Functional Description 1. Enhances security products and provides system intelligence. The same vulnerability data warehouse information is shared with the products so that more intelligent actions may then be taken to mitigate complex threats. 2. Cross-references the threat's identifier with the target's configuration. The threat's CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's existing OS, application, and patch configuration profile. An answer is returned accurately indicating if the machine is vulnerable. 3. Identifies network vulnerabilities. The platform will independently identify which network machines and devices have vulnerabilities by querying a client agent, or a device's OS or firmware directly. 4. Proactively remediates the vulnerability. Once a vulnerability is identified, a remediation function is provided that may remotely and automatically deploy the appropriate update to each machine or device, install it, and document it with change tracking and audit trail reports.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 10: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be integrated into network security products for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment – Remotely remediates network vulnerabilities and provides policy compliance and enforcement capabilities. See <i>SecurityProfiling's</i> SysUpdate white paper and trial download online. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix upgrades. 2. IDS – Accurately determines if the targeted machine is actually vulnerable to an attack; virtually eliminates false positives and false negatives; and adds vulnerability remediation and policy compliance and enforcement functionalities. (See <i>SecurityProfiling's</i> Intelligent IDS application.)

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 15: Solution Anti-Vulnerability technology will integrate with and enable network security products to intelligently reference and share information from the same vulnerability data set, provide vulnerability identification and remediation functionalities, and efficiently meet policy compliance and enforcement requirements. The Anti-Vulnerability platform is a complimentary network security technology. When integrated into the defense strategy, it adds intelligence that more accurately and efficiently mitigates blended threats and offloads the time-consuming functions that burden network administrators. The platform enables network security products to share information via its proprietary logic engine to automatically cross-reference the threat identifier with the targeted machine's configuration to determine if it is actually vulnerable to that threat. Previously separate and uncoordinated processes are now more intelligent and automated, resulting in improved system accuracy and efficiency. Therefore the need to layer more and more point products and add a SIMS is reduced, in-turn decreasing the amount of data – particularly erroneous data – to manage. Subsequently, the costs to acquire, operate, and manage the additional multiple point products, and the need to implement a SIMS, are also reduced. The platform will also remotely and proactively remediate vulnerabilities by first determining which updates are needed and compatible with each machine or device, taking into account the OS, applications, or firmware installed. Then, the updates may be deployed, installed, and validated. Thereby, policy compliance is effectively and efficiently enforced, and documented.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 19: Anti-Vulnerability™ Technology Applications Examples of Anti-Vulnerability applications that may be made available for enterprise and small office home office (SOHO) networks when integrated with Anti-Vulnerability technology via the SDK: 1. Update or Patch Deployment <i>SecurityProfiling's</i> SysUpdate patch management and vulnerability remediation solution is the first Anti-Vulnerability technology application. A white paper and trial download are available online that demonstrate its Anti-Vulnerability functionalities and benefits. Benefits: Anti-Vulnerability technology enables products to add accurate vulnerability identification, remediation, verification, and policy compliance functions. With Anti-Vulnerability technology, products will gain intelligence, accuracy and efficiency, eliminate false positives and false negatives, and ensure policy compliance, thereby saving the organization time and money. Such functionalities and benefits are available for both the enterprise and SOHO networks. And, for the consumer market, it will additionally support non-security related updates such as application version, performance, and bug-fix updates for widely distributed programs, which will optimize system performance and enhance the consumer experience.

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 27-28: 4. Patch Management Functions • GetPatchInfo – Parameters: PATCH to get information about, PATCHINFO pointer to receive information Returns: Success or failure Description: This function fills the passed patch info structure with information about the passed patch • InstallPatchForClient – Parameters: CLIENT to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to the client. Installation proceeds asynchronously • InstallPatchForGroup – Parameters: GROUP to install the patch on, PATCH to install Returns: Success or failure Description: This function deploys the patch to all clients in the group • InstallAllPendingForClient – Parameters: CLIENT to install patches on Returns: Success or failure Description: This function deploys all pending patches for a client to the client • InstallAllPendingForGroup – Parameters: GROUP to install patches on Returns: Success or failure Description: This function deploys all pending patches to all clients in a group • RejectPatchForClient – Parameters: CLIENT to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for a client. This patch will not be queued down or installed

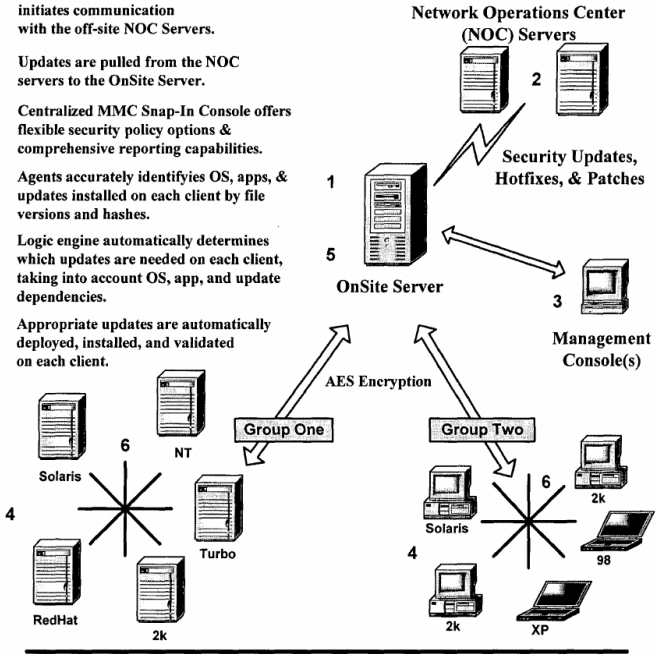
Claim	Exemplary Citation from [REFERENCE]
	• RejectPatchForGroup – Parameters: GROUP to reject patch for, PATCH to reject Returns: Success or failure Description: This function rejects a patch for every client in a group. This patch will not be queued down or installed • RequeuePatchForClient – Parameters: CLIENT to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for a client • RequeuePatchForGroup – Parameters: GROUP to requeue the patch for, PATCH to requeue Returns: Success or failure Description: This function requeues a previously rejected patch for every client in a group • RemovePatchFromClient – Parameters: CLIENT to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch from a client • RemovePatchFromGroup – Parameters: GROUP to remove patch from, PATCH to remove Returns: Success or failure Description: This function will uninstall a patch for every client in a group • InstallServicePackForClient – Parameters: CLIENT to install service pack on, SERVICEPACK to install on client Returns: Success or failure Description: This function will install a service pack on a client • InstallServicePackForGroup – Parameters: GROUP to install service pack on, SERVICEPACK to install Returns: Success or failure Description: This function will install a service pack on every client in a group • InstallSoftwareForClient – Parameters: CLIENT to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on a client • InstallSoftwareForGroup – Parameters: GROUP to install software update on, SOFTWARE to install Returns: Success or failure Description: This function will install a software update on every client in a group. • GetCveIDsFromPatch – Parameters: PATCH to get CveIDs from Returns: A LIST of CVEIDs Description: This function returns a list of CVEIDs the correspond to the PATCH

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 33: transform via the Anti-Vulnerability SDK, which will add the following functions. Functional Description 1. Cross-references the threat's identifier with the target's configuration. The CVE ID, or other identifier, and the Destination IP address are fed into the logic engine where it cross-references the threat with the machine's configuration profile. 2. Virtually eliminates false positives and false negatives. The backend accurately determines in real time if the targeted machine is susceptible to the attack. And, if the machine is not susceptible, it is filtered and reported back as an event and not an incident. No further data or alert is generated, and a management response is not required. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and document it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines if targeted machine is vulnerable to an attack in real time • Filters out erroneous, benign and false positive alerts • Remotely patches targeted vulnerabilities in one click • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	<p data-bbox="596 235 940 264">Oliphant Provisional at 35:</p> <div data-bbox="625 293 709 315"> <p>Solution</p> </div> <p data-bbox="625 321 982 412">SecurityProfiling offers a new intelligent application called SysUpdate — the first technology to fully automate the updating and reporting processes, as described below.</p> <div data-bbox="695 436 909 457"> <p>Functional Description</p> </div> 1. Automatically researches updates. In collaboration with software development companies, SysUpdate receives updates and their respective compatibility and installation guidelines, which are thoroughly reviewed and tested in a lab for system compatibility and stability. 2. Automatically and securely downloads, and archives, all updates. Once the updates and guidelines are thoroughly reviewed and tested, they are automatically downloaded to each customer's OnSite Server, verified with MD5 hashes, and archived in a self-populating patch repository database. 3. Automatically determines the update and dependency requirements for all computers on a network, based on their individual operating systems, applications, and previously installed updates. The system's logic engine automatically matches the tested updates and their guidelines across each computer's configuration profile. 4. Remotely, securely, and automatically deploys updates to each computer and device. Updates may be tested on a test group, and then queued for release to each computer on the network, consistently in accordance with the policy determined by the network administrator. 5. Automatically verifies the updates are installed and running correctly. Each computer's client agent installs the updates, and verifies the files and hashes. 6. Automatically generates change tracking and audit trail reports. Change tracking and audit trail reports may be selected and printed from the Management Console's Crystal Reports modules. Additionally, application license audit reports may be generated, which lists software installed on each machine, by license key. <div data-bbox="1073 318 1232 339" data-cs="2" data-kind="parent"> <p>Unique Features:</p> </div> • Supports Windows, Linux variants, and network devices • Supports non-Microsoft applications • Logic engine provides system intelligence • Files verified with SHA1 or MD5 hashes • MMC snap-in console with Crystal Reports • AES encryption

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 41: Functional Description 1. Cross-references the threat's identifier with the target's configuration. The Inline Sensor monitors and processes traffic and sends alert data to the Anti-Vulnerability OnSite Server where its logic engine queries the backend in real-time to determine if the destination IP is vulnerable to the attack. 2. Enhanced Flex Response. If the destination IP is vulnerable to the attack, the Inline Sensor is commanded to immediately drop the exploit packets – preventing the attack. 3. Remotely remediates the vulnerability. When a machine is identified as vulnerable to an attack, an incident, a remediation function is also provided to the administrator to remotely deploy the appropriate update to the machine or device, install it, verifies the files and hashes, and documents it with change tracking and audit trail reports. Unique Features: • Supports Windows and UNIX variants • Determines in real-time if targeted machine is vulnerable to an attack, virtually eliminating false positives • Inline Sensor flex response terminates the attack • Remotely patches targeted vulnerabilities in one shot • Installs in minutes

Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 42: Intelligent IPS Configured Network Overview Diagram The diagram illustrates the architecture of an Intelligent IPS Configured Network. At the top, 'Off-Site Network Operations Center (NOC) Servers' (labeled 2) are connected to an 'OnSite Server' (labeled 1 and 5) via 'AES Encryption'. The OnSite Server is also connected to an 'IPS Console'. The OnSite Server (1) sends 'Security Rules, Updates & Patches' to the IPS Console. The OnSite Server (5) is connected to an 'Inline Sensor' (labeled 3a and 3b). The Inline Sensor is connected to a 'Perimeter Firewall'. The Perimeter Firewall is connected to a 'Hub'. The Hub is connected to two groups of client machines. The left group includes 'Solaris', 'NT', 'RedHat', and 'Turbo' (labeled 4 and 6). The right group includes 'Mandrake', '2k', 'Solaris', and '98' (labeled 4 and 6). The diagram is divided into two main sections by a horizontal line, with the top section showing the server and console components and the bottom section showing the client machines and network infrastructure. 1 The OnSite Server periodically initiates communication with the NOC Servers. 2 Updates are pulled from the NOC Servers to the OnSite Server. 3a IPS Inline Sensor monitors and processes network traffic, and sends real time alert data to the OnSite Server where it cross-references the data with the Anti-Vulnerability backend. 3b If the destination IP is vulnerable to the attack, the Inline Sensor is commanded in real-time to drop the malicious packets. 4 Agents are deployed to each client, accurately identifying which OS, apps, & updates are installed. 5 Logic engine automatically determines which updates are needed to remediate the targeted machine, and may be remotely deployed from the IPS Cosole. 6 Appropriate updates are remotely deployed, installed, and validated on each client.

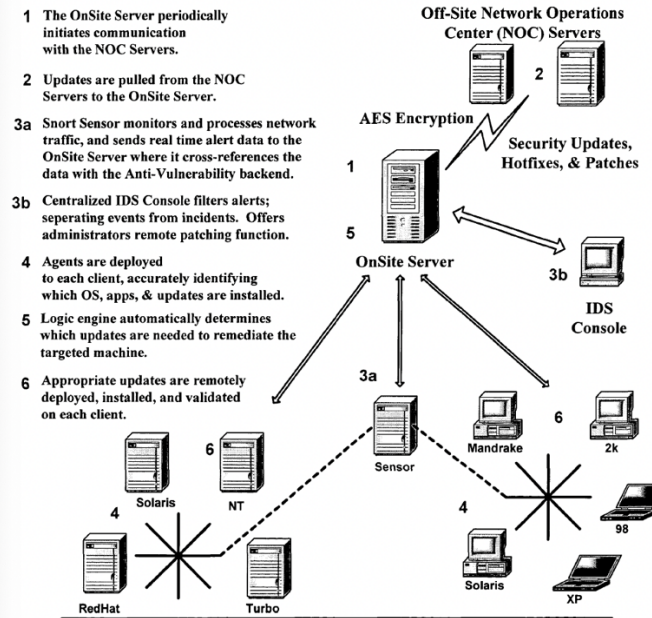
Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 36: SysUpdate™ Configured Network Overview Diagram  1 The OnSite Server periodically initiates communication with the off-site NOC Servers. 2 Updates are pulled from the NOC servers to the OnSite Server. 3 Centralized MMC Snap-In Console offers flexible security policy options & comprehensive reporting capabilities. 4 Agents accurately identifies OS, apps, & updates installed on each client by file versions and hashes. 5 Logic engine automatically determines which updates are needed on each client, taking into account OS, app, and update dependencies. 6 Appropriate updates are automatically deployed, installed, and validated on each client. Network Operations Center (NOC) Servers 2 Security Updates, Hotfixes, & Patches 1 5 OnSite Server 3 Management Console(s) AES Encryption Group One Group Two 4 4 6 Solaris NT Turbo RedHat 2k Solaris 2k 98 2k XP

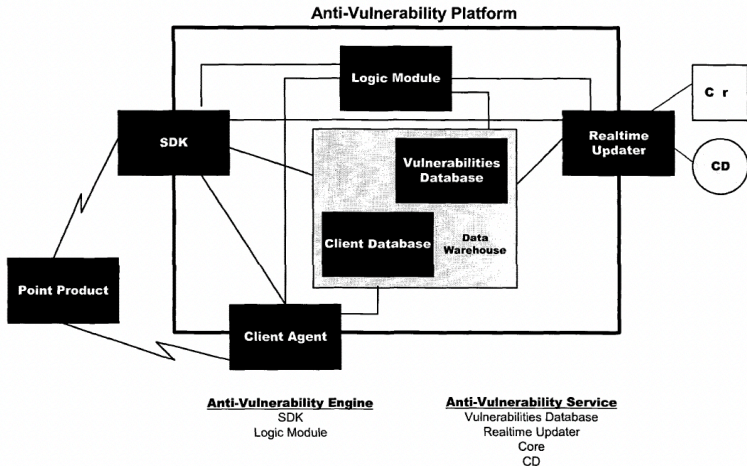
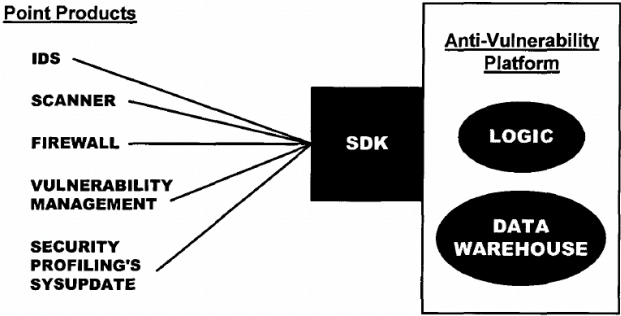
Claim

Exemplary Citation from [REFERENCE]

Oliphant Provisional at 34:

Intelligent IDS Configured Network Overview Diagram



Claim	Exemplary Citation from [REFERENCE]
	Oliphant Provisional at 16: Figure 2 – SDK and Platform Component Diagram  Oliphant Provisional at 12: Figure I – Anti-Vulnerability Technology  <i>For more detail, see the Platform Component Diagram on page 8.</i>

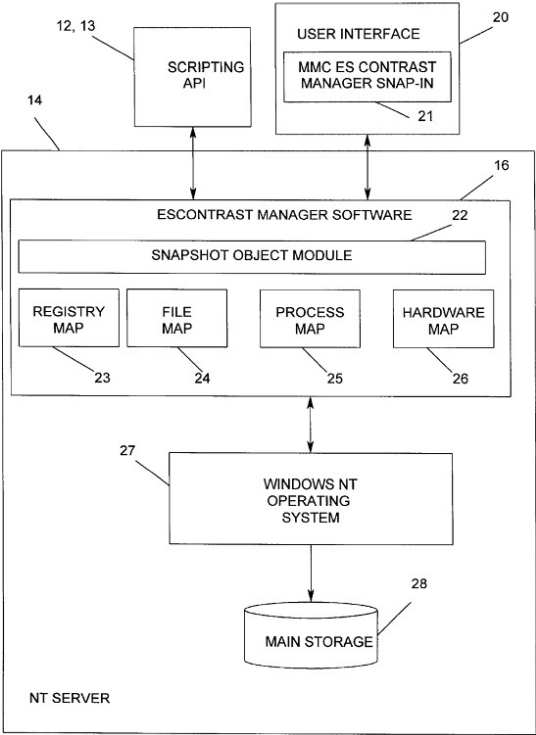
Claim	Exemplary Citation from [REFERENCE]
	Largman at 19:29: "An automated process may repair a data storage device of a computer system. The repair process may include multiple programs. The automated process may be triggered by a particular event or a set of events. The repair process may be specific to a particular data storage device such as the primary boot partition of a hard drive. The repair process may encompass a variety of functions which may be modified, added, or skipped based on the type of repair or user preferences. The user may modify user preferences. In one embodiment, the repair process represents a sequence of functions. Typically a Master Template is either provided to the user or created by the user. Backups are created intermittently. The computer system becomes unstable and repair becomes necessary. The user may activate the repair process or the repair process may recognize the instability or problems with the system and activate the repair process. Prior to repair, a Master Template typically exists for the computer system. The Master Template may have been created in a number of different ways. Several ways of creating one or more Master Templates for this computer system include: shipped with a new computer, created with the installation of software (e.g., software to support this process), created by a user-activated program, periodically created of a Master Template by a program. Backups typically exist for a computer system. A backup may include user data and programs which have been stored on a data storage device accessible to the computer system. For example, documents may have been created or modified by a user. These documents may be stored as a backup. The user may have installed additional programs that may be stored in a backup. During a backup process data is copied from a data storage device of the computer system to the backup data storage device(s). Any data that is accessible to the computer system may be backed up. The backup may be compressed. Compression may reduce the amount of storage space required to hold the backup. Incremental backups may also be used. Incremental backups may reduce the time required to perform a backup and reduce the storage space required to store them. Backups may be stored as archives." Largman at 35:64: "Optionally, a special-purpose subsystem may be repaired or returned to an ideal state using an automated repair process. Such repairs may be conducted "on the fly", or after each transaction or without rebooting. Master templates typically represent an ideal state of a special-purpose subsystem, and may be stored on a storage system. A transaction may include reading e-mail, wherein the opening of each individual e-mail messages represents a separate transaction. Optionally, one or

Claim	Exemplary Citation from [REFERENCE]
	more items can be ignored during a repair process. For example if an e-mail has been opened, a repair process may run ignoring the open e-mail, detect and repairs problems, and then a user may respond to the e-mail without quitting it. In another embodiment, all downloads and e-mail can be saved immediately to the storage system prior to opening the download or e-mail in the work subsystem. In one embodiment, the logic of a special-purpose subsystem, such as a working system, may trigger an event associated with a repair process. The repair process may perform a comparison between a master template of the working system and state of the current working system. Any differences between them could trigger a subsequent repair process in which some or all data that is different is deleted from the working system. Further, data may be copied from the master template by the repair process as necessary. In one embodiment, the repair process may make the working system identical to the master template. In one embodiment, a repair process can be conducted after one or more e-commerce transactions, or after surfing one or more web pages, and the like. Thus all known and unknown viruses and Trojan horses can be made impotent prior to the next transaction. While this process does not eliminate viruses, worms and Trojan horses from the computer (they may be stored in the storage system), it keeps them in an in operative state. The repair process could repair volatile and non-volatile memory, or clear volatile memory, or set volatile memory to an ideal state. In one embodiment, if the user selects more than one e-mail to open, two or more e-mails could be copied to the working system and could be open simultaneously. Optionally each e-mail could be copied to its own separate isolated working system, opened, viewed, and worked on separately. If the user needs to copy data from one isolated e-mail to another isolated e-mail, a copying process can be used that does not allow code to execute. In one embodiment, web commerce software, or e-mail software, or any software can be modified so that individual records, or only copies of the records that are specifically needed for a transaction are copied to the storage system, utilized and then copied back to the database in the storage system, and after each such transaction a repair can be conducted. Optionally, in a transaction in which data interacts with more than one database or CGI for example, the transaction can be broken up into discrete segments, data copied to and from the isolated storage system(s) or working systems as needed, and repairs can be run between each segment of a transaction, or between some segments of a transaction. Optionally, software can contain instructions that define what type of data can comprise a transaction, limiting the copy process to only copying data that meets certain criteria."

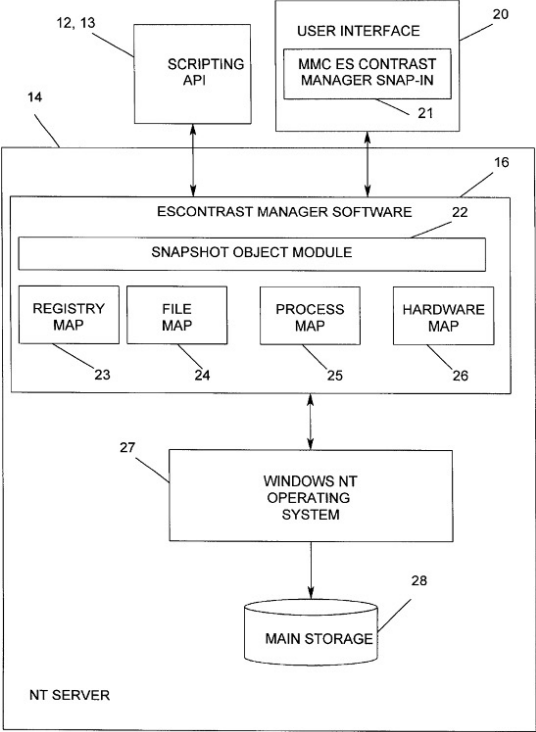
Claim	Exemplary Citation from [REFERENCE]
	Gertner at 26:48: "In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them. Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In

Claim	Exemplary Citation from [REFERENCE]
	any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remedial actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states. Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art."
[7] The method of claim 1, wherein individual snapshots include data associated with at least one of: system files, application files, a registry entry, a performance counter, a process, a communication port, a hardware configuration, a log file, a running task, services, and network connections.	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards, printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components." Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided." Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond, Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27."

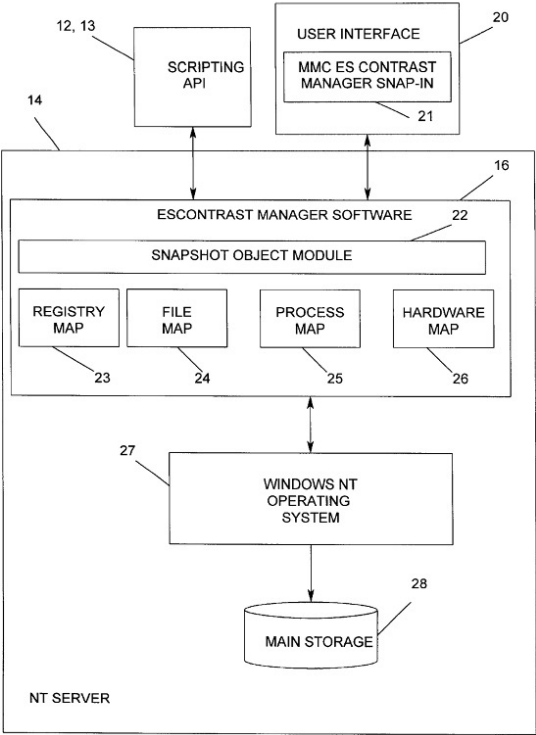
Claim	Exemplary Citation from [REFERENCE]
	Evoy at Fig. 2 and corresponding text:  <pre> graph TD subgraph NT_SERVER [NT SERVER] subgraph Top_Controls [14] Scripting_API[12, 13: SCRIPTING API] User_Interface[20: USER INTERFACE 21: MMC ES CONTRAST MANAGER SNAP-IN] end subgraph ESContrast_Manager_Software [16: ESCONTRAST MANAGER SOFTWARE] Snapshot_Object_Module[22: SNAPSHOT OBJECT MODULE] Registry_Map[23: REGISTRY MAP] File_Map[24: FILE MAP] Process_Map[25: PROCESS MAP] Hardware_Map[26: HARDWARE MAP] end Windows_NT_OS[27: WINDOWS NT OPERATING SYSTEM] Main_Storage[(28: MAIN STORAGE)] Scripting_API <--> ESContrast_Manager_Software User_Interface <--> ESContrast_Manager_Software ESContrast_Manager_Software <--> Windows_NT_OS Windows_NT_OS --> Main_Storage end </pre> FIG. 2
[8] The method of claim 1, further comprising: inserting rules into the rule set of the adaptive reference model to augment or override rules of the rule set automatically generated from the snapshots.	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards, printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs

Claim	Exemplary Citation from [REFERENCE]
	some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information." Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components." Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided." Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond,

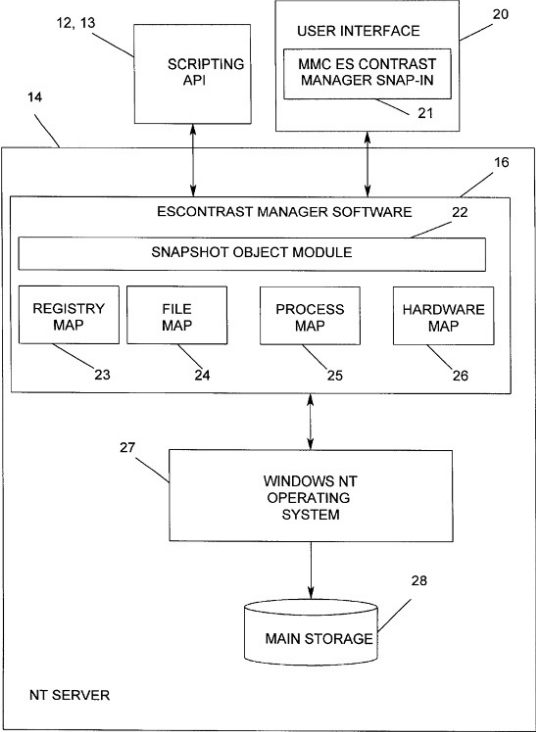
Claim	Exemplary Citation from [REFERENCE]
	Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27." Evoy at Fig. 2 and corresponding text:  FIG. 2
[9] The method of claim 1, wherein the adaptive reference model is generated to include a value	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards,

Claim	Exemplary Citation from [REFERENCE]
layer that determines whether an asset value contained in a snapshot is anomalous.	printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information." Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components." Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond, Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27."

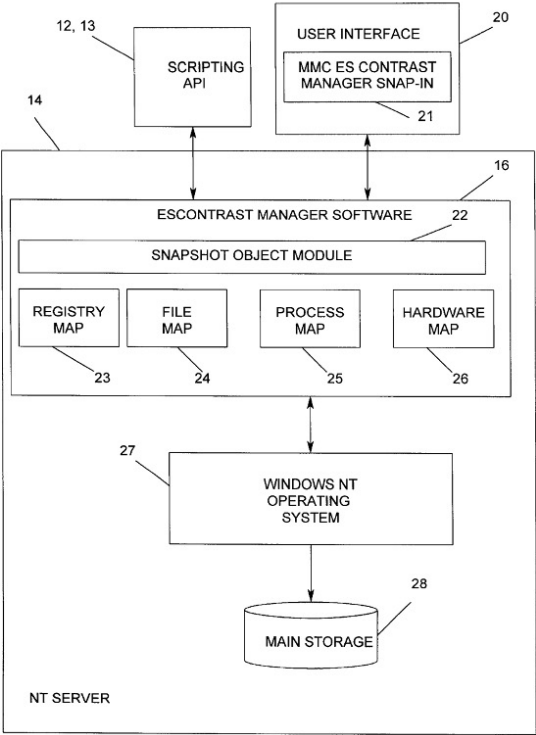
Claim	Exemplary Citation from [REFERENCE]
	Evoy at Fig. 2 and corresponding text:  FIG. 2
[10] The method of claim 1, wherein the adaptive reference model is generated to include a cluster layer that tracks relationships between assets and identifies an anomaly in response to an asset being unexpectedly absent from or	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards, printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs

Claim	Exemplary Citation from [REFERENCE]
present in a set of assets in a snapshot.	some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information." Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components." Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided." Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond,

Claim	Exemplary Citation from [REFERENCE]
	Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27." Evoy at Fig. 2 and corresponding text:  FIG. 2
[11] The method of claim 1, wherein the adaptive reference model is generated to include a	Evoy at 1:25: "The components of a computer system that may be of interest at run-time are its hardware, file systems, processes running on the system and its registry. The term hardware of a computer system refers to objects that can be actually touched like disks, disk drives, display screens, keyboards,

Claim	Exemplary Citation from [REFERENCE]
profile layer that identifies anomalies in response to violation of relationships of clusters of assets in a snapshot.	printers, boards, and chips. In contrast software is untouchable. Software exists as ideas, concepts, and symbols but has no substance. File system refers to a system that an operating system or a program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. A process is a program being executed by a computer system and the term is used loosely as a synonym for task hereinbelow. A process performs some useful operations on data. Typically, an operating system is capable of running several processes at the same time. Hence, at any given time there could be one or more processes running on a computer system. A registry is a database used by the Windows (also Windows NT) operating system to store configuration information." Evoy at 2:41: "It is therefore an object of this invention to provide a method for comparing and contrasting System States from different points in time in order for a user to diagnose and solve complex system administration problems in a time span shorter than that provided with the prior art techniques. Users of an embodiment of the present invention would be typically computer technology administrators, researchers, or developers who would be interested in comparing the states of the components of a computer system at different points in time. By using this invention they will be able to diagnose system problems more efficiently, both in terms of speed and accuracy. This would result in overall cost reductions in operating and maintaining a computer system as well as higher availability levels of its critical components." Evoy at 3:22: "These and other objects, which will become apparent as the invention is described in detail below, where in a computer system having a memory and an operating system persistently stored in a storage device, a method is disclosed for comparing System States of the system from time to time. The method includes creating a snapshot object for interfacing with functionalities of the system; and using the snapshot object, creating a first collection of System State objects which record the status of the system at a given instant in time. Next, the first collection of System State objects is stored in the memory. A second collection of System State objects, which was previously created and stored in the storage device, is then loaded into the memory. A comparison of the first collection with the second collection is made and an indicia of any differences therebetween is provided."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at 5:9: "Referring now to FIG. 2, a modular diagram representing the system and method of the present invention is shown. The Windows NT server 14 includes the ES Contrast Manager software 16, Windows NT Operating System 26, and a shared main storage 27. The scripting API 12 or 13, or an MMC Contrast Manager snap-in 21, which is available from Microsoft Corporation of Redmond, Washington, may be used to access the Contrast Manager Software 16. The Contrast Manager software 16 includes a snapshot object module 22, a Registry Map module 23, a File Map module 24, a Process Map module 25, and a Hardware Map module 26. The Contrast Manager software 16 runs on the Windows NT operating system 27."

Claim	Exemplary Citation from [REFERENCE]
	Evoy at Fig. 2 and corresponding text:  FIG. 2 The diagram illustrates the architecture of an NT Server. At the top, a 'SCRIPTING API' (12, 13) and a 'USER INTERFACE' (20) containing an 'MMC ES CONTRAST MANAGER SNAP-IN' (21) are shown. These components interact with the 'ESCONTRAST MANAGER SOFTWARE' (22) within the 'NT SERVER' (14). The 'ESCONTRAST MANAGER SOFTWARE' (22) includes a 'SNAPSHOT OBJECT MODULE' (16) and four sub-modules: 'REGISTRY MAP' (23), 'FILE MAP' (24), 'PROCESS MAP' (25), and 'HARDWARE MAP' (26). This software layer interacts with the 'WINDOWS NT OPERATING SYSTEM' (27), which in turn connects to 'MAIN STORAGE' (28).
[12PRE] A computer readable medium storing instructions, that when executed by a computer, cause the computer to perform functions of:	<i>See [1PRE].</i>
[12A] receiving snapshots from a plurality of computers within a	<i>See [1A].</i>

Claim	Exemplary Citation from [REFERENCE]
population of computers, wherein individual snapshots include data indicating a state of a respective computer;	
[12B] automatically generating an adaptive reference model comprising a rule set customized to characteristics of the population of computers, the rule set being developed by identifying patterns among the snapshots from the plurality of computers such that the adaptive reference model is indicative of normal states in the computers within the population; and	<i>See</i> [1B].
[12C] comparing a snapshot from at least one of the computers to the adaptive reference model to determine whether an anomaly is present in the state of the at least one of the computers.	<i>See</i> [1C].
[13] The computer readable medium of claim 12, further comprising instructions, that when executed, cause the	<i>See</i> Claim 2.

Claim	Exemplary Citation from [REFERENCE]
computer to compare the anomaly to a recognition filter to diagnose a trouble condition on the at least one of the computers, and, in the event of a trouble condition, to generate an automated response to the trouble condition.	
[14PRE] The computer readable medium of claim 13, wherein the automated response is a generic response not specific to a particular asset of the at least one computer, and wherein the computer readable medium further comprises instructions, that when executed, cause the computer to:	<i>See</i> [3PRE].
[14A] send the generic response and a set of anomalies found in the snapshot to the at least one computer, the set of anomalies indicating assets of the at least one computer whose states are anomalous; and	<i>See</i> [3A].
[14B] apply the generic response to the anomalous	<i>See</i> [3B].

Claim	Exemplary Citation from [REFERENCE]
assets to correct the trouble condition.	
[15] The computer readable medium of claim 14, wherein the generic responses include at least one of: installing a missing software component, removing an undesirable software component, and restoring an incorrect registry setting.	<i>See Claim 4.</i>
[16] The computer readable medium of claim 13, wherein the recognition filter comprises a particular pattern of anomalies that indicates the presence of a particular root cause condition or a generic class of conditions.	<i>See Claim 5.</i>
[17] The computer readable medium of claim 12, wherein the instructions for comparing the snapshot to the adaptive reference model include instructions for determining whether an asset value contained in a snapshot is anomalous.	<i>See Claim 9.</i>
[18] The computer readable medium of claim 12,	<i>See Claim 10.</i>

Claim	Exemplary Citation from [REFERENCE]
wherein the instructions for comparing the snapshot to the adaptive reference model include instructions for tracking relationships between assets and identifying an anomaly in response to an asset being unexpectedly absent from or present in a set of assets in a snapshot.	
[19] The computer readable medium of claim 12, wherein the instructions for comparing the snapshot to the adaptive reference model include instructions for identifying anomalies in response to violation of relationships of clusters of assets in a snapshot.	<i>See</i> Claim 11.
[20PRE] The computer readable medium of claim 12, wherein the computer readable medium further comprises instructions, that when executed, cause the computer to:	<i>See</i> [6PRE].
[20A] compare a plurality of anomalies associated with a particular snapshot with a recognition filter to	<i>See</i> [6A].

Claim	Exemplary Citation from [REFERENCE]
diagnose a trouble condition; and	
[20B] diagnose a trouble condition on the at least one of the computers in response to at least a subset of the plurality of anomalies matching information in the recognition filter.	<i>See</i> [6B].
[21] The computer readable medium of claim 12, wherein individual snapshots include data associated with at least one of: system files, application files, a registry entry, a performance counter, a process, a communication port, a hardware configuration, a log file, a running task, services, and network connections.	<i>See</i> Claim 7.
[22] The computer readable medium of claim 12, wherein the computer readable medium further comprises instructions, that when executed, cause the computer to: insert rules into the rule set of the adaptive reference model to augment or override rules of	<i>See</i> Claim 8.

Claim	Exemplary Citation from [REFERENCE]
the rule set automatically generated from the snapshots.	