(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 360 071 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
30.12.2020 Bulletin 2020/53

(21) Application number: **16854157.1**

(22) Date of filing: **04.10.2016**

(51) Int Cl.:
G06F 21/57 (2013.01) *G06F 21/60* (2013.01)
H04L 29/06 (2006.01) *G06F 21/56* (2013.01)

(86) International application number:
PCT/US2016/055273

(87) International publication number:
WO 2017/062338 (13.04.2017 Gazette 2017/15)

(54) **METHOD AND SYSTEM FOR IDENTIFICATION OF SECURITY VULNERABILITIES**

VERFAHREN UND SYSTEM ZUR ERKENNUNG VON SICHERHEITSSCHWACHSTELLEN

PROCÉDÉ ET SYSTÈME D'IDENTIFICATION DE VULNÉRABILITÉS DE SÉCURITÉ

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **06.10.2015 US 201514876592**

(43) Date of publication of application:
15.08.2018 Bulletin 2018/33

(73) Proprietor: **Assured Enterprises, Inc.**
Reston, Virginia 20190 (US)

(72) Inventor: **LI, David**
Reston, Virginia 20190 (US)

(74) Representative: **Barker Brettell LLP**
100 Hagley Road
Edgbaston
Birmingham B16 8QQ (GB)

(56) References cited:
WO-A2-2011/068967
US-A1- 2004 064 726
US-A1- 2008 244 691
US-A1- 2011 321 164
US-A1- 2012 222 123
US-A1- 2013 333 032
US-A1- 2014 123 279
US-B1- 7 845 007
US-B1- 7 845 007

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

WIZ, Inc. EXHIBIT - 1013
WIZ, Inc. v. Orca Security LTD.

Description

TECHNICAL FIELD OF THE INVENTION

[0001] Embodiments of the present invention relate generally to security of electronic devices and systems and, more particularly, to identification of security vulnerabilities

BACKGROUND

[0002] Computer systems may include many elements communicatively coupled to one another via a network. Networking and sharing of elements adds a level of complexity that is not present with a single element that stands alone. Network and system administrators may manage network elements using various software tools, which may include a graphical user interface.

[0003] Application code may run on computer systems. One application may have code running on various elements of a computer system. The application itself may be managed by network or system administrators using various software tools.

[0004] Malware may attack computer systems. Malware may include spyware, rootkits, password stealers, spam, sources of phishing attacks, sources of denial-of-service-attacks, viruses, loggers, Trojans, adware, or any other digital content that produces malicious activities. Furthermore, an application may be vulnerable to malware or other exploitative attacks

[0005] Application WO2011068967 discloses a malware analysis method, performing code scans based on analyses of import tables to identify called application code functions/components .

SUMMARY

[0006] In one embodiment, a system for securing an electronic device includes a processor and a memory. The memory may be communicatively coupled to the processor and include instructions. The instructions, when loaded and executed by the processor, cause the processor to scan data including one or more application components to uniquely identify elements therein, determine from a given application component additional components to be accessed by the given application component, scan the additional components to uniquely identify elements therein, determine whether the additional components include any known vulnerabilities, associate one or more known vulnerabilities of the additional components with the given application component, record the known vulnerabilities and the given application component. The given application component may be uniquely identified.

[0007] In another embodiment, a machine readable storage medium may include computer-executable instructions that are readable by a processor. The instructions, when read and executed, may be for causing the processor to scan data including one or more application components to uniquely identify elements therein, determine from a given application component additional components to be accessed by the given application component, scan the additional components to uniquely identify elements therein, determine whether the additional components include any known vulnerabilities, associate one or more known vulnerabilities of the additional components with the given application component, record the known vulnerabilities and the given application component. The given application component may be uniquely identified.

[0008] In yet another embodiment, a method of securing an electronic device may include scanning data including application components to uniquely identify elements therein, determining from a given application component additional components to be accessed by the given application component, scanning the additional components to uniquely identify elements therein, determining whether the additional components include any known vulnerabilities, associating one or more known vulnerabilities of the additional components with the given application component, and recording the known vulnerabilities and the given application component. The given application component may be uniquely identified.

[0009] In one embodiment, a system may include a memory. The memory may be communicatively coupled to the processor and include instructions. The instructions, when loaded and executed by the processor, cause the processor to identify one or more application components uniquely identified and determine vulnerabilities associated with a given application component. The vulnerabilities may include vulnerabilities of additional components to be accessed by the given application component. The processor may be caused to adjust characterizations of the vulnerabilities associated with the given application component based upon contextual information from the system in which the given application component resides. The contextual information may include security information.

[0010] In another embodiment, a machine readable storage medium may include computer-executable instructions that are readable by a processor. The instructions, when read and executed, may be for causing the processor to identify one or more uniquely identified application components and determine vulnerabilities associated with a given application component. The vulnerabilities may include vulnerabilities of additional components to be accessed by the given application component. The processor may be caused to adjust characterizations of the vulnerabilities associated with the given application component based upon contextual information from the system in which the given application component resides. The contextual information may include security information.

[0011] In yet another embodiment, a method may include identifying one or more application components uniquely identified and determining vulnerabilities asso-

ciated with a given application component. The vulnerabilities may include vulnerabilities of one or more additional components to be accessed by the given application component. The method may include adjusting characterizations of the vulnerabilities associated with the given application component based upon contextual information from the system in which the given application component resides. The contextual information may include security information.

BRIEF DESCRIPTION OF THE FIGURES

[0012] For a more complete understanding of the configurations of the present disclosure, needs satisfied thereby, and the objects, features, and advantages thereof, reference now is made to the following description taken in connection with the accompanying drawings.

FIGURE 1 is a block diagram of an example system for identifying security vulnerabilities, in accordance with the teachings of the present disclosure;

FIGURE 2 is an illustration of example operation and further configuration of the system for identifying security vulnerabilities, in accordance with the teachings of the present disclosure;

FIGURE 3 is an illustration of further example operation of the system for identifying security vulnerabilities, in accordance with the teachings of the present disclosure; and

FIGURE 4 is a flow chart of an example method for identifying security vulnerabilities, in accordance with the teachings of the present disclosure.

DETAILED DESCRIPTION

[0013] FIGURE 1 is an illustration of an example embodiment of a system 100 for identifying security vulnerabilities, in accordance with the teachings of the present disclosure. System 100 may include any suitable number and kind of elements. For example, system 100 may include one or more devices that can identify security vulnerabilities by scanning electronic devices, file systems, Java applications, .NET applications, or other sources of electronic data. Such scanning may be performed locally to the source of electronic data or remotely on another electronic device communicatively coupled through a network to the source of electronic data. For example, system 100 may include one or more agents 102 configured to scan sources of electronic data for vulnerabilities. In another example, system 100 may include a server 104 configured coordinate scanning sources of electronic data for vulnerabilities. System 100 may include any suitable number and kind of source of electronic data, such as files or file system 114, that may be scanned for vulnerabilities. Although file system 114 is shown separate from any clients or servers, file system 114 may be resident on the same device as client 102 or server 104.

[0014] Server 104 may be configured to coordinate scanning of various sources of information by agents 102. Server 104 may be implemented in any suitable manner, including by one or more applications, scripts, libraries, modules, code, drivers, or other entities on an electronic device. These may include software or instructions resident on a memory 124 for execution by a processor 122. Although sever 104 is illustrated in FIGURE 1 as including example elements, server 104 may include more or less elements. Moreover, the function of some elements of server 104 as discussed herein may be performed in various embodiments by other elements of server 104. Also, the function of some elements of server 104 as discussed herein may be performed in various embodiments by elements of client 102. For example, server 104 may include a communication application 120, security enterprise manager 126, update manager 134, scan scheduler 128, policy manager 130, or a central repository 132.

[0015] Client 102 may be configured scan various sources of information such as file system 114. Client 102 may be implemented in any suitable manner, including by one or more applications, scripts, libraries, modules, code, drivers, or other entities on an electronic device. These may include software or instructions resident on a memory 118 for execution by a processor 116. Although client 102 is illustrated in FIGURE 1 as including example elements, client 102 may include more or less elements. Moreover, the function of some elements of client 102 as discussed herein may be performed in various embodiments by other elements of client 102. For example, client 102 may include a communication application 110, scan application 108, and local repository 112. Client 102 may communicate with server 104 through network 106.

[0016] Client 102 and server 104 may communicate with sources of information about vulnerability of software. Any suitable sources of information may be utilized by client 102 and server 104. For example, server 104 may communicate with one or more vulnerability databases 138, 140. Database 138 may be a publicly accessible vulnerability database, while database 140 may be a proprietary vulnerability database. Although a single such database 138, 140 is shown and described, multiple public or proprietary databases may be accessed. Database 138 may include the National Vulnerability Database (NVD). Database 138 may include a repository of standards-based vulnerability management data. The database may further include databases of security checklists, security related software flaws, misconfigurations, product names, product versions, exploitability metrics, impact metrics, temporal metrics, environmental metrics, and others. Server 104 may communicate with a system evaluation database 136, which may include information about the overall health of a system in which file system 114 (or other data under evaluation) resides. Each of these databases may be implemented in any suitable manner, such as by a relational database, navigational database, or other organization of data and data

structures. Server 104 may integrate the contents from these databases to provide comprehensive coverage of known vulnerabilities.

[0017] Communication application 120 and communication application 110 may be configured to handle inbound and outbound communications to other entities for server 104 and client 102. For example, communication application 120 and communication application 110 may handle communications with file system 114, databases 138, 140, 126, and between server 104 and client 102. Communication application 120 and communication application 110 may be implemented by any suitable mechanism, such as an application, function, library, application programming interface, script, executable, code, software, or instructions. These may in turn be implemented by instructions resident in memory for execution by a processor that, when loaded into the processor, cause the functionality described in this disclosure to be performed.

[0018] Security enterprise manager 126 may be configured to organize scanning operations in system 100. Security enterprise manager 126 may determine, for example, what agents 102 need to scan their respective sources of data, how agents 102 will scan, how information will be reported from agents 102, what remedial action might be taken or recommended, when agents 102 will be updated, and other such configurations and operations of system 100. Security enterprise manager 126 may utilize a scan scheduler 128 to determine or dictate how often and under what conditions scans of data will be made and repeated. Furthermore, security enterprise manager 126 may utilize an update manager 134 to determine or dictate how often and under what conditions information to be used by scan application 108 will be updated. Update manager 134 may be configured to gather information from one or more sources about how to scan data, such as database 138, 140, 136. Update manager 134 may be configured to store relevant information to be used by agents 102 in central repository 132. Contents from central repository 132 may be selectively provided to agents 102 by update manager. Security enterprise manager 126 may utilize a policy manager 130 configured to analyze the overall health of a system under evaluation. Policy manager 130 may be configured to access information from, for example, system evaluation database 136. Security enterprise manager 126, update manager 134, scan scheduler 128, and policy manager 130 may be implemented by any suitable mechanism, such as an application, function, library, application programming interface, script, executable, code, software, or instructions. These may in turn be implemented by instructions resident in memory for execution by a processor that, when loaded into the processor, cause the functionality described in this disclosure to be performed.

[0019] Scan application 108 may be configured to scan data under evaluation in system 100. The data may be located on the same electronic device as scan application 108 or on an electronic device communicatively coupled to scan application 108. Scan application may analyze the data under evaluation to determine whether the data indicates any vulnerabilities to users of the data. Scan application may utilize a local repository 112 to hold rules, guidelines, settings, or other data collected by server 104. Local repository 112 may be implemented by any suitable manner of implementing databases or other data structures. Scan application 108 may be configured to scan data, such as those in file system 114, at any appropriate time. Scan application 108 may be implemented by any suitable mechanism, such as an application, function, library, application programming interface, script, executable, code, software, or instructions. These may in turn be implemented by instructions resident in memory for execution by a processor that, when loaded into the processor, cause the functionality described in this disclosure to be performed.

[0020] In operation, scan application 108 may search for holes, vulnerabilities, or other possible exploitations in software. Such software may include files in file system 114. Scan application may look for signatures of software binaries that are defined in local repository 112. Such signatures may be imported from original sources, such as databases 138, 140. Scan application 108 may search and scan software located on a given computer, desktop, smartphone, tablet, or other suitable electronic device. In some embodiments, scan application 108 may search and scan a defined installation image that is to be installed on multiple clients. Scan application 108 may identify files or subcomponents or files in file system 114 that have been identified as having a vulnerability. In some embodiments, such a file might not be malicious itself, but may be exploitable by malware.

[0021] The memories may be in the form of physical memory or pages of virtualized memory. The processors may comprise, for example, a microprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), or any other digital or analog circuitry configured to interpret and/or execute program instructions and/or process data. In some embodiments, the processor may interpret and/or execute program instructions and/or process data stored in memory. Memory may be configured in part or whole as application memory, system memory, or both. Memory may include any system, device, or apparatus configured to hold and/or house one or more memory modules. Each memory module may include any system, device or apparatus configured to retain program instructions and/or data for a period of time (e.g., computer-readable storage media). Instructions, logic, or data for configuring the operation of the system may reside in memory for execution by the processor. Program instructions may be used to cause a general-purpose or special-purpose processing system that is programmed with the instructions to perform the operations described above. The operations may be performed by specific hardware components that contain hardwired logic for performing the operations, or by any

combination of programmed computer components and custom hardware components.

[0022] FIGURE 2 is an illustration of operation of system 100 and of further configuration thereof, in accordance with embodiments of the present disclosure. Scan application 108 may be scanning a sequence of files on file system 114. Scan application 108 may encounter a given file, such as XYZ.exe 202. Scan application 108 may determine a unique identification of the file. In one embodiment, scan application 108 may determine the actual contents of XYZ.exe 202 by determining a signature, hash, or other unique digital identifier of XYZ.exe 202. The unique identification may precisely identify the version, build, or other particular instance of XYZ.exe, of which there may be many versions or completely different sources.

[0023] Scan application 108 may check whether the signature of XYZ.exe 202 matches any known software elements populated in local repository 112. If there are any known vulnerabilities of XYZ.exe 202 noted in local repository 112, they may be noted. The entries of local repository 112 may be marked or indexed according to a hash, signature, or other identifier. Moreover, the vulnerabilities may be categorized or defined by a unique identifier, so that consumers of the results from scan application 108 may efficiently apply its results.

[0024] Many files might not be known to be safe or vulnerable, as myriad different software creators create myriad different pieces of software. Accordingly, in one embodiment scan application 108 might not find an indication of XYZ.exe 202 in local repository 112. The existing binary signature and use of scanning of top-level applications may be of little use. As shown in FIGURE 1, there might not be an entry for XYZ.exe 202 therein. Alternatively, there may be an entry denoting that XYZ.exe has no known vulnerabilities. Based upon either such case, in one embodiment scan application 108 might determine that XYZ.exe 202 itself has no known vulnerabilities.

[0025] However, in some embodiments a file might make use of still other files. For example, a file might access other files by calling external functions. These external functions might be executed in, for example, a shared library. The compiled binaries of the shared library may be statically or dynamically linked, included, or otherwise associated with the binaries of the original file. For example, XYZ.exe 202 may be dynamically linked to a DLL such as ABC.dll 204. Many applications might be linked to, share, and use such a file.

[0026] In one embodiment, scan application 108 may determine the set of libraries or other external code that are to be accessed by a given file. Scan application 108 may analyze the software and application file structure to identify such components. An application executable may be in a portable executable format. The format may include a data structure that contains the information needed for the operating system loader to manager the wrapped executable code. The file format may begin with a header that specifies information about the code in the file, the type of application, required library headers, and space requirements. The header may further specify an import table that identifies functions used by the file to access external components and the locations of such functions. Scan application 108 may parse this information to determine what external components, libraries, or other entities that XYZ.exe 202 executes, such as ABC.dll 204. Any suitable file format or structure may be parsed and analyzed by scan application 108 to determine the wrappings or packaging to identify external components used by the file.

[0027] In some cases, required components such as shared library may be stored as separate files in the file system. In other cases, required components may be embedded in the executables themselves. When additional required components are stored as separate files, some file-based scanners cannot associate the identified vulnerabilities to the correct executables that ultimately use the required components. When required components are embedded in the executables themselves, some file-based scanners will miss the executables as-incorporating the required components because the binary signatures do not exist for the executable files as they exist while incorporating these required components. However, in either case scan application 108 may identify the types of applications or executables based on a specific operating system, file extensions, other file attributes, and the file signatures. The file signatures may include hex codes around the beginning of the files, known as "magic numbers. Based on the types of applications, the executable file structures can be known, as well as the required components. From these, potential vulnerable system calls may be identified. For example, an application .EXE file (an executable application with file extension "exe" on Windows™ Operating Systems) may use the Portable Executable (PE) file format. The PE file format is a data structure that contains the information necessary for the Windows™ Operating System loader to manage the wrapped executable code. The PE file format begins with a header that includes information about the code, the type of application, required library functions, and space requirements. Furthermore, the import table of the PE file header contains the information about specific functions used by this executable and the locations of these functions. For Linux/Unix-based executables, scan application 104 may use the dynamic loader of the system to examine the dynamic section of an executable to identify all needed components, such as shared libraries used by given dynamically-linked executables.

[0028] Scan application 108 may in turn scan these libraries based upon the determination. For example, scan application 108 may scan ABC.dll 204 after finding that it is linked to XYZ.exe 202. Moreover, in another embodiment the scan results of the linked library may be ascribed to the original file. For example, even if XYZ.exe itself was determined to have no vulnerabilities in local

repository 112, the vulnerabilities of ABC.dll 204 denoted in local repository 112 may be subsequently associated with XYZ.exe.

[0029] Furthermore, as shown in FIGURE 2, local repository 112 may have entries that are specific to individual versions of files. The vulnerabilities of different versions of the same file may be different. The vulnerabilities, directly or indirectly attributed to XYZ.exe, may be reported by scan application 108.

[0030] FIGURE 3 illustrates further example operation of system 100, in accordance with embodiments of the present disclosure. Upon completion of all or some of the scan of file system 114, scan application 108 may report vulnerabilities to other parts of system 100 for corrective action or reporting. In one embodiment, system 100 may utilize policy manager 130 to determine what corrective action or reporting is to be performed.

[0031] System 100 may handle different determined vulnerabilities in different ways. In one embodiment, system 100 may identify that a component has a vulnerability that is of a qualitatively or quantitatively higher or lower priority based upon the degree of malicious behavior available for malware. Such a component may include a file. In another embodiment, system 100 may identify that a component has a particular vulnerability, but that vulnerability is enhanced or lessened by other aspects of the system in which the file resides.

[0032] System evaluation database 136 may include a complete diagnostic evaluation of the system in which the file resides. For example, system evaluation database 136 include information about the candidate system, including whether it has firewalls, where such firewalls are located, what kind of firewalls exist, and how they are configured. In one embodiment, the interplay of the vulnerabilities of the files as determined by scan application 108 and the candidate system conditions from system evaluation database 136 may yield whether or not corrective action is necessary. In another embodiment, a prioritization of the corrective action or vulnerabilities may be produced. System evaluation database 136 may include information about the candidate system with any suitable number of permutations and combinations of security software, hardware, or settings thereof resident on the candidate system.

[0033] For example, if the particular instance of ABC.dll is known to be vulnerable to a particular network-based exploit, the vulnerability may be added to an evaluation of XYZ.exe 202. Security enterprise manager 128 may receive the vulnerability list. Security enterprise manager 128 may consult system evaluation database 136 to evaluate the system that included XYZ.exe 202 with respect to its configuration. Policy manager 128 may interpret an entry in system evaluation database 126 that the candidate system (for example, DEF) has a particular firewall installed with given settings. The settings of the particular firewall may be tuned to defeat the particular vulnerability identified in XYZ.exe 202 through its use of ABC.dll. Consequently, security enterprise manager 126 may rank the known vulnerability of XYZ.exe 202 qualitatively or quantitatively less than other vulnerabilities.

[0034] In some embodiments, security enterprise manager 128 may produce or infer a composite vulnerability measurement. The composite vulnerability measurement may be inferred from the vulnerability of identified individual components. For example, the composite vulnerability severity measurement may be a weighted aggregate of individual severity measurements. For example, base, temporal, and environmental factors may be used. The weights associated with vulnerabilities of each factor may be customized, or industry-standard weights may be used. Each of these factors may include a subgroup of related factors. Furthermore, each group may produce a composite quantitative score with a vector used to illustrate the components used to derive the score.

[0035] In one embodiment, through policy manager 128 and system evaluation database 136, security enterprise manager 128 may contextualize standard metrics received from databases 138, 140 based upon specific characteristics of the candidate system. For example, as discussed above, the firewall used in the candidate system may be considered. In another embodiment, policy manager 128 may characterize the security operating environment of the candidate system using a three-dimensional mathematical model. In various embodiments, the three dimensions may include threat vector, threat impact, and enterprise maturity of cyber defense. A threat vector may refer to the path or mechanism that potential attackers employ to gain access to a computer, a system, or an enterprise. Identification of the vectors provides the insights of how the attackers (or other threat agents) exploit the associated vulnerabilities. The threat impact may refer to the impact when those vulnerabilities are successfully exploited. There are a wide range of different impacts that may be categorized or quantified, including but not limited to data loss, confidentiality compromise, loss of revenue, damages to the systems, etc. In some embodiments, the ranking of a specific vulnerability will be boosted if it associated with an application that in turn is associated with any critical enterprise asset. The enterprise maturity may refer to a wide range of enterprise assessments, including organization architecture, networks software/application, computing infrastructure wireless, intrusion detection/prevention, access control, security policy, risk assessment, data flow, configure management, enterprise application, data integrity, enterprise monitoring, physical security, contingency management, operating environment, and backup service. In the end, a composite score may be given to measure the enterprise cyber maturity.

[0036] While these enterprise characteristics may be manifest themselves by impacting standard scoring as described above, they may be uniquely considered together. For example, from system evaluation database 136 may include information from an audit that security controls may have been in place in DEF such as firewalls,

intrusion detection systems, intrusion prevention systems. These components may affect metrics regarding threat access, access complexity, and target distribution. For example, the presence of a modern firewall might make the access difficult and increase the complexity of the manner of attacking because the attackers have to determine a more complex manner of bypassing the firewall to be able to successfully exploit the identified vulnerability. The target distribution may refer to the proportion of the impacts to the vulnerable system due to a specific identified vulnerability. For example, firewalls may be used to establish multiple security zones. If one security zone is compromised, the proportion of impact would be limited to that security zone.

[0037] In another example, the location of the vulnerability, such as the machine on which it resides, may be included in evaluating the vulnerability if it is co-located with identified critical enterprise assets. In such a case, the vulnerability risk or threat level may be increased. The contextualization of the threat may be increased or decreased by applying specific weights in response to these determinations. The weight might be applied to individual metrics. The individual metrics may include Access Vector, Access Complexity, Authentication, Confidential Impact, Integrity Impact, Availability Impact, Exploitability, Remediation Level, Report Confidence, Collateral Damage Potential, Target Distribution, Confidentiality, Integrity Requirement, and Availability Requirement. The access vector may reflect how the vulnerability is exploited. The access complexity may measure the complexity of the means required to exploit the vulnerability. The authentication may measure the number of times an attacker must authenticate to a target in order to exploit a vulnerability. An associated weight would provide additional insights on strength and complexity of the authentication process. The confidentiality impact may measure the impact on confidentiality of a successfully exploited vulnerability. The integrity impact may measure the impact to integrity of a successfully exploited vulnerability. The availability impact may measure the impact to availability of a successfully exploited vulnerability. The exploitability may measure how easy or available the attack is for malicious agents, such as the availability of easy-to-use exploit code. The remediation level may measure how urgent vulnerability needs to be dealt with due to the patch level. For example, the remediation level may be quantified based upon whether a temporary fix or workaround is available. The report confidence may measure the degree of confidence in the existence of a vulnerability. The collateral damage potential may measure the risk of loss of life or other physical assets. This may be typically related to what kind of business is served by the systems, as well as any emergency response policy and procedures that are in place. The target distribution may measure the proportion of impact of vulnerability system. The confidentiality, integrity, and availability requirement may measure the importance of the affected enterprise assets. The identification and classification of enterprise assets directly impact these measurements through weights or scores.

[0038] In one embodiment, security enterprise manager 126 may evaluate multiple vulnerabilities resulting from the file system and provide a ranked list of the vulnerabilities discovered during scanning. A given scan may yield hundreds or thousands of vulnerabilities, and corrective action for each may be expensive. The ranking may be used to prioritize patching of the most critical vulnerabilities first. When comparing the threat levels posed by different vulnerabilities, any suitable combination of factors may be considered. For example, the number of vulnerabilities for a given file, the severity of the vulnerabilities, and a confidence of identification of vulnerability may be used. A ranking score may be used that is a weighted aggregate of the individual measurements of each factor. These vulnerability measurements are further combined to produce the ranking of the identified vulnerabilities. The ranking is critical to enable enterprise prioritizing the limited resources to patch most critical vulnerabilities first. Multiple factors are involved in the consideration of ranking, for example, the number of vulnerability counts, the severity of vulnerability, and the confidence of identified vulnerability, etc. For example, if one shared component has more vulnerabilities associated with it or more application uses than another, then quantified counts of vulnerabilities or application uses would impact the ranking or importance of the component in the context of prioritizing remediation. The ranking score is generated as a weighted aggregate of the individual measurement of each factor.

[0039] Scan application 108 may run any suitable number of times. Scan scheduler 128 may dictate that a given file system may be scanned during specified, periodic, or customized times. Each scan operation may be performed independently from one another. Repeated scans at different times may yield additional insights through temporal correlation. For example, scanning the same target after a first scan and taking corrective action on identified vulnerabilities may provide an indication of the success of the corrective action. Security enterprise manager 126 may analyze successive scans and identify whether the corrective actions to be taken were successful.

[0040] For example, the corrective action to fix XYZ.exe 202 may be specified as an upgrade to the latest version. However, as the problem lied with ABC.dll 204, the upgrade to XYZ.exe 202 might not have fixed the problem and XYZ.exe 202 might still be vulnerable because of its use of ABC.dll. Security enterprise manager 126 may determine such a continued problem after a second scan and identify different remedial action, noting the previous attempt that failed to solve the problem. In another example, the new version of XYZ.exe 202 may itself have new vulnerabilities. Thus, the initial scan and corrective action may be insufficient to fully secure the candidate system. In yet another example, since a previous scan a new threat may have been identified to an

existing version of XYZ.exe 202, or to a recently patched version of XYZ.exe 202. In still yet another example, the nature of an existing threat may have changed or evolved since a previous scan. Examples of such evolution may include cases where an effective exploit might have become more widely available, thus the potential number of attackers may have grown and the vulnerability is thus more sever. Accordingly, its ranking may need to be increased. In another case, a remediation may have become ineffective, particularly when a remediation was a temporary fix. Similarly, a more effective remediation may have been determined, and thus a vulnerability more easily patched. Successive scans may identify one or more of these examples and security enterprise manager 126 may change the vulnerability determinations accordingly. Accordingly, system 100 may provide protection based not only upon vulnerability but the responsiveness of counter-measures. In one embodiment, the responsiveness of counter-measures may be used as a factor for the contextualization discussed above.

[0041] In one embodiment, scanning may be repeated based upon the persistence or level of threats. For example, scan scheduler 128 may schedule repeated scans after a severe threat is attempted to be remediated. Furthermore, scan scheduler 128 may schedule a repeat scan for only those portions of the candidate system that have been changed with the corrective action. In another embodiment, an incremental scan may be used to speed up the scanning process. For example, security enterprise manager 126 may cause scanning of newly updated or changed parts of a candidate system. Scan application 108 may monitor a client system and report new changes. An incremental scan might be made for the newly identified changes to, for example, registry, application directories, and the file system. By limiting the scan to these elements, it might be performed more quickly.

[0042] Frequent scanning of an entire candidate system may be expensive in terms of time and resources. In one embodiment, scan scheduler 128 may take into account threat scores for a given system as represented in system evaluation database 136 to determine whether or not to perform a scan, or how often to perform a scan. For example, if system DEF includes a high threat score with respect to any two of the three factors of enterprise maturity, threat vector, and threat impact. An integral risk model [may specify an overall risk that the system may pose from the environmental factors, independent of specific application vulnerabilities for which scan application 108 may search. For example, there may be critical assets associated with the system or cyber defense measures associated with the system. These considerations as reflected in the assessment score model may constitute an integral risk model which may help to determine the recommended scan schedule. The costs of scanning may be weighed against this threat score by scan scheduler 128 to determine how often to scan.

[0043] In one embodiment, security enterprise manager 126 may take into account other scanning results in determining vulnerabilities. For example, security enterprise manager 126 may access or initiate scanning of operating system configurations or network devices. Such elements may be within the same system as the file system 114. The scans may determine settings or vulnerabilities within the host environment in which a candidate file exists. The results of such scans may be stored in system evaluation database 136. For example, a file might have a vulnerability, but its application may be configured to require an authentication before it can be exploited. Scanning the operating system or network settings may inform security enterprise manager 126 whether the operating system has been misconfigured such that the authentication for the file is not required, and thus the vulnerability might be worse.

[0044] In one embodiment, system evaluation database 136 may provide recommended remediation in the form of penetration testing, or pentesting. The suggested pentesting may be based upon the determined vulnerabilities from scanning and their associated severity. For example, consider that a given media player may be prone to various heap-based buffer-overflow, allowing attackers to execute arbitrary code to cause serious damages and data losses. These vulnerabilities may cause severe client-side exploitation risks, as the attacks may focus on client-side software applications such as web browser and email applications. In another example, the "heartbleed" vulnerability may be identified for a software widget, wherein an OpenSSL cryptographic software library vulnerability, widely used in applications, may allow stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure many part of the Internet. This vulnerability allows anyone from the outside to read memory of the enterprise system and compromises secret keys used for encryption. In response to such determined vulnerabilities from scanning, security enterprise manager 126 may identify additional software, modules, scripts, or corrective measures to be taken to perform pentesting on the identified vulnerabilities. Penetration testing applications may be loaded and the candidate system may be attacked to determine whether the candidate system is vulnerable to the identified attack. The penetration testing applications could include, for example, Metasploit Framework.

[0045] In another embodiment, security enterprise manager 126 may identify or create intrusion detection and intrusion protection rules from a knowledge base. These may be tailored to an identified vulnerability and configured to facilitate attack detection and protection. For example, once a media player vulnerability is identified, security enterprise manager 126 may identify rules to be applied to security software, wherein the rules are tailored to the vulnerability. For example, snort rule updates related to media player vulnerabilities may include specific protections against a zero length assertion heap overflow attempt, convolution filter heap information disclosure attempt, and byte array double free attempt. Each of these might be a different vector of attack upon the

same media player. The rules may match the particular threat vector used to exploit the vulnerability. The rules may be accessed from, for example, a Snort intrusion prevention system.

[0046] FIGURE 4 is a flow chart of an example method 400 for identifying security vulnerabilities, in accordance with the teachings of the present disclosure. Method 400 may be implemented on any suitable device or system, such as those illustrated in FIGURES 1-3. Method 400 may include any suitable number or order of steps, and may begin at any suitable point. The steps of method 400 may be optionally repeated, whether entirely or individually. Moreover, method 400 may be performed by more or fewer steps than those illustrated below. Method 400 may be initiated by any suitable input. During performance, method 400 may return to previously performed steps or jump to other steps not yet performed. Various portions of method 400 may be performed in parallel with each other as needed.

[0047] In one embodiment, method 400 may begin at 405.

[0048] At 405, systems to be evaluated may be audited. The systems may be characterized and given dimensions of evaluations and scores. Information about applications and software may be obtained from third party databases. Repositories of information may be updated at client scanners. The time and scope of scanning of a given client may be scheduled.

[0049] At 410, using a determined scope of scan-such as a range of applications-, the system may be scanned. The scan may uniquely identify files and applications and match them to known values of vulnerabilities. Additional libraries or other components that may be accessed by the file being scanned may in turn be scanned. The structure of the file may be analyzed to determine what additional libraries or other components to scan.

[0050] At 415, vulnerabilities of the file may be recorded. Furthermore, vulnerabilities of the file may be associated with vulnerabilities of the libraries. The vulnerabilities may be reported to a server.

[0051] At 420, the vulnerabilities may be evaluated and augmented in view of contextual information about the environment of the system on which the file resides. Threat scores of the system may be determined and incorporated into the vulnerabilities. At 425, the vulnerabilities may be ranked or prioritized with respect to one another.

[0052] At 430, suitable corrective measures to correct for vulnerabilities may be identified, such as reinstallation, upgrades, changing settings, pentesting, or new instruction rules. At 435, corrective measures may be selectively applied according to their severity and the cost of the corrective measure. Some corrective measures may be postponed or omitted.

[0053] At 440, it may be determined whether a scan will be repeated for elements associated with changes made in 435. If so, method 400 may repeat at 410 for the changed elements. Otherwise, at 445 it may be determined whether a scan period has ended, wherein scans will be repeated. If so, method 400 may repeat at 410. Otherwise, at 450 it may be determined whether changed conditions, such as in the nature of threats or of the system, require rescan. If so, method 400 may repeat at 410 for the elements associated with the changes. Otherwise, at 455 it may be determined whether method 400 should terminate. If not, method 400 may repeat at, for example, 445. Otherwise method 400 may terminate.

[0054] Method 400 may be implemented fully or in part by instructions on computer-readable media for execution by a processor. For the purposes of this disclosure, computer-readable media may include any instrumentality or aggregation of instrumentalities that may retain data and/or instructions for a period of time. Computer-readable media may include, without limitation, storage media such as a direct access storage device (e.g., a hard disk drive or floppy disk), a sequential access storage device (e.g., a tape disk drive), compact disk, CD-ROM, DVD, random access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM), and/or flash memory; as well as communications media such wires, optical fibers, and other electromagnetic and/or optical carriers; and/or any combination of the foregoing.

[0055] Although the present disclosure has been described in detail, it should be understood that various changes, substitutions, and alterations can be made hereto without departing from the scope of the disclosure as defined by the appended claims.

Claims

1. A system for electronic security, comprising:

a processor; and

a memory communicatively coupled to the processor and including instructions, the instructions, when loaded and executed by the processor, cause the processor to:

analyze, during a first scan, an application file structure of an application and an import table of the application to identify, via the application file structure of the application and the import table of the application, one or more uniquely identified application components;

determine vulnerabilities associated with a given application component of the one or more uniquely identified application components, the vulnerabilities including vulnerabilities of one or more additional components to be accessed by the given application component;

adjust characterizations of the vulnerabilities associated with the given application

component based upon contextual information from the system in which the given application component resides, the contextual information including security information; and

take remedial action based upon the adjusted vulnerability characterizations; and

after the remedial action is taken on the given application component:

perform a subsequent scan of the given application component after a remedial action is taken on the given application component, based on a determination that a threat associated with the vulnerability has changed;

determine, based on the subsequent scan of the given application component, that the remedial action corrected at least one of the vulnerabilities associated with the given application component, present before the remedial action is taken on the given application component;

determine, based on the subsequent scan of the given application component, one or more new vulnerabilities associated with the given application component;

determine from an interplay of the vulnerabilities of the given application component, that includes the one or more new vulnerabilities, that corrective action is necessary; and

adjust a firewall device to defeat at least one of the one or more new vulnerabilities for the corrective action, wherein the firewall device is used to establish multiple security zones.

2. The system of Claim 1, wherein the memory further includes instructions for causing the processor to:

determine a measure of the vulnerabilities; and rank the vulnerabilities; or

determine a measure of the vulnerabilities including a composite score from a plurality of vulnerability factors; and rank the vulnerabilities.

3. The system of Claim 1, wherein the memory further includes instructions for causing the processor to identify pentesting suggestions to further confirm presence of vulnerabilities based upon the identity of the vulnerabilities.

4. The system of Claim 1, wherein, to take remedial action, the memory further includes instructions for causing the processor to take the remedial action in the form of changed intrusion detection or protection rules based upon the identity of the vulnerabilities.

5. The system of Claim 1, wherein adjusting the characterizations of the vulnerabilities is further based upon:

a security configuration of the system related to the vulnerabilities; or

contextual information including threat vectors, threat impacts, and enterprise maturity of the system; or

at least upon two or more of an access vector, access complexity, authentication, confidential impact, integrity impact, availability impact, exploitability, remediation level, report confidence, collateral damage potential, target distribution, confidentiality, integrity requirement, availability requirement, or

a determined change in a threat causing the vulnerability.

6. The system of Claim 1, wherein the one or more new vulnerabilities includes multiple new vulnerabilities.

7. The system of Claim 1 wherein adjusting the network security device to defeat the at least one of the one or more new vulnerabilities for the corrective action includes adjusting the network security device to defeat a first new vulnerability of the new vulnerabilities and omit a second new vulnerability of the new vulnerabilities.

8. A method of electronic security, comprising

analyzing, during a first scan, an application file structure of an application and an import table of the application to identify, via the application file structure of the application and the import table of the application, one or more uniquely identified application components;

scanning the one or more uniquely identified application components to determine vulnerabilities associated with a given application component of the one or more uniquely identified application components, the vulnerabilities including vulnerabilities of one or more additional components to be accessed by the given application component;

adjusting characterizations of the vulnerabilities associated with the given application component based at least upon contextual information from the system in which the given application component resides, the contextual information including security information; and

taking remedial action based upon the adjusted vulnerability characterizations; and

after the remedial action is taken on the given application component:

performing a subsequent scan of the given application component after a remedial action is taken on the given application component, based on a determination that a threat associated with the vulnerability has changed;
determining, based on the subsequent scan of the given application component, that the remedial action corrected at least one of the vulnerabilities associated with the given application component, present before the remedial action is taken on the given application component;
determining, on the subsequent scan of the given application component, one or more new vulnerabilities associated with the given application component;
determining from an interplay of the vulnerabilities of the given application component, that includes the one or more new vulnerabilities, that corrective action is necessary; and
adjusting a firewall device to defeat at least one of the one or more new vulnerabilities for the corrective action, wherein the firewall device is used to establish multiple security zones.

9. The method of claim 8, further including:

determining a measure of the vulnerabilities; and rank the vulnerabilities; or
determining a measure of the vulnerabilities including a composite score from a plurality of vulnerability factors; and rank the vulnerabilities.

10. The method of Claim 8, further comprising identifying pentesting suggestions to confirm a presence of the vulnerabilities based upon the identity of the vulnerabilities.

11. The method of Claim 8, further comprising performing remedial action in the form of changed intrusion detection or protection rules based upon the identity of the vulnerabilities.

12. The method of Claim 8, wherein adjusting the characterizations of the vulnerabilities is further based upon:

a security configuration of the system related to the vulnerabilities; or
contextual information including threat vectors, threat impacts, and enterprise maturity of the system; or
at least upon two or more of an access vector, access complexity, authentication, confidential impact, integrity impact, availability impact, exploitability, remediation level, report confidence, collateral damage potential, target distribution, confidentiality, integrity requirement, availability requirement, or
a determined change in a threat causing the vulnerability.

13. The method of Claim 8, wherein the one or more new vulnerabilities includes multiple new vulnerabilities.

14. The method of Claim 8, wherein adjusting the network security device to defeat the at least one of the one or more new vulnerabilities for the corrective action includes adjusting the network security device to defeat a first new vulnerability of the new vulnerabilities and omit a second new vulnerability of the new vulnerabilities.

15. At least one machine readable storage medium, comprising computer-executable instructions carried on the machine readable storage medium, the instructions readable by a processor, the instructions, when read and executed, for causing the processor to perform the method of any of Claims 8to 14.

Patentansprüche

1. System für elektronische Sicherheit, umfassend:

einen Prozessor; und
einen Speicher, der kommunikativ an den Prozessor gekoppelt ist und Anweisungen beinhaltet, wobei die Anweisungen, wenn geladen und durch den Prozessor ausgeführt, den Prozessor veranlassen zum:

Analysieren, während eines ersten Scans, einer Anwendungsdateistruktur einer Anwendung und einer Importtabelle der Anwendung, um, über die Anwendungsdateistruktur der Anwendung und die Importtabelle der Anwendung, eine oder mehrere einzigartig identifizierte Anwendungskomponenten zu identifizieren;
Ermitteln von Schwachstellen, die mit einer bestimmten Anwendungskomponente der einen oder mehreren einzigartig identifizierten Anwendungskomponenten verknüpft sind, wobei die Schwachstellen Schwachstellen einer oder mehrerer zusätzlicher Komponenten beinhalten, einen Zugriff durch die gegebene Anwendungskomponente zu erfahren;
Einstellen von Charakterisierungen der Schwachstellen, die mit der gegebenen Anwendungskomponente verknüpft sind, basierend auf Kontextinformationen von dem System, in dem sich die gegebene Anwendung befindet, wobei die Kontextinformationen Sicherheitsinformationen beinhalten;

und
Ergreifen einer Behebungsmaßnahme, basierend auf den eingestellten Schwachstellencharakterisierungen; und
nach Ergreifen der Behebungsmaßnahme an der gegebenen Anwendungskomponente:

Durchführen eines folgenden Scans der gegebenen Anwendungskomponente nach Ergreifen einer Behebungsmaßnahme bei der gegebenen Anwendungskomponente, basierend auf einer Bestimmung, dass sich eine Bedrohung, die mit der Schwachstelle verknüpft ist, geändert hat;
Bestimmen, basierend auf dem folgenden Scan der gegebenen Anwendungskomponente, dass die Behebungsmaßnahme mindestens einer der Schwachstellen behoben hat, die mit der gegebenen Anwendungskomponente verknüpft sind, die vor Ergreifen der Behebungsmaßnahme an der gegebenen Anwendungskomponente vorhanden waren;
Bestimmen, basierend auf dem folgenden Scan der gegebenen Anwendungskomponente, einer oder mehrerer neuer Schwachstellen, die mit der gegebenen Anwendungskomponente verknüpft sind;
Bestimmen, aus einem Zwischenspiel der Schwachstellen der gegebenen Anwendungskomponente, die die eine oder mehreren neuen Schwachstellen beinhaltet, dass eine Korrekturmaßnahme erforderlich ist; und
Einstellen einer Firewall-Vorrichtung, um mindestens eine der einen oder mehreren neuen Schwachstellen für die Korrekturmaßnahme zu bekämpfen, wobei die Firewall-Vorrichtung verwendet wird, um mehrere Sicherheitszonen zu errichten.

2. System nach Anspruch 1, wobei der Speicher weiter Anweisungen beinhaltet, um den Prozessor zu veranlassen zum:

Bestimmen eines Ausmaßes der Schwachstellen; und Reihen der Schwachstellen; oder
Bestimmen eines Ausmaßes der Schwachstellen, beinhaltend einen zusammengesetzten Punktewert aus einer Vielzahl von Schwachstellenfaktoren; und Reihen der Schwachstellen.

3. System nach Anspruch 1, wobei der Speicher weiter Anweisungen beinhaltet, um den Prozessor zu veranlassen, Pentest-Vorschläge zu identifizieren, um weiter ein Vorhandensein von Schwachstellen basierend auf der Identität der Schwachstellen zu bestätigen.

4. System nach Anspruch 1, wobei zum Ergreifen einer Behebungsmaßnahme der Speicher weiter Anweisungen beinhaltet, um den Prozessor zu veranlassen, die Behebungsmaßnahme in der Form geänderter Eindringerfassungs- oder Schutzregeln zu ergreifen, basierend auf der Identität der Schwachstellen.

5. System nach Anspruch 1, wobei Einstellen der Charakterisierungen der Schwachstellen weiter basiert auf
einer Sicherheitskonfiguration des Systems, die sich auf die Schwachstellen bezieht; oder
Kontextinformationen, die Bedrohungsvektoren, Bedrohungseinflüsse und Unternehmensreife des Systems beinhalten; oder
mindestens auf zwei oder mehr von einem Zugriffsvektor, Zugriffskomplexität, Authentifizierung, Vertrauenseinfluss, Integritätseinfluss, Verfügbarkeitseinfluss, Nutzbarkeit, Behebungsniveau, Berichtsicherheit, Kollateralschadenpotential, Zielverteilung, Vertraulichkeit, Integritätsanforderung und Verfügbarkeitsanforderung oder
einer bestimmten Änderung in einer Bedrohung, die die Schwachstelle verursacht.

6. System nach Anspruch 1, wobei die eine oder mehreren neuen Schwachstellen mehrere neue Schwachstellen beinhalten.

7. System nach Anspruch 1, wobei Einstellen der Netzwerksicherheitsvorrichtung zum Bekämpfen der mindestens einen der einen oder mehreren neuen Schwachstellen für die Behebungsmaßnahme Einstellen der Netzwerksicherheitsvorrichtung beinhaltet, um eine erste neue Schwachstelle der neuen Schwachstellen zu bekämpfen und eine zweite neue Schwachstelle der neuen Schwachstellen wegzulassen.

8. Verfahren für elektronische Sicherheit, umfassend:

Analysieren, während eines ersten Scans, einer Anwendungsdateistruktur einer Anwendung und einer Importtabelle der Anwendung, um, über die Anwendungsdateistruktur der Anwendung und die Importtabelle der Anwendung, eine oder mehrere einzigartig identifizierte Anwendungskomponenten zu identifizieren;
Scannen der einen oder mehreren einzigartig identifizierten Anwendungskomponenten zum

Ermitteln von Schwachstellen, die mit einer bestimmten Anwendungskomponente der einen oder mehreren einzigartig identifizierten Anwendungskomponenten verknüpft sind, wobei die Schwachstellen Schwachstellen einer oder mehrerer zusätzlicher Komponenten beinhalten, einen Zugriff durch die gegebene Anwendungskomponente zu erfahren;
Einstellen von Charakterisierungen der Schwachstellen, die mit der gegebenen Anwendungskomponente verknüpft sind, basierend mindestens auf Kontextinformationen von dem System, in dem sich die gegebene Anwendung befindet, wobei die Kontextinformationen Sicherheitsinformationen beinhalten; und
Ergreifen einer Behebungsmaßnahme, basierend auf den eingestellten Schwachstellencharakterisierungen; und
nach Ergreifen der Behebungsmaßnahme an der gegebenen Anwendungskomponente:

Durchführen eines folgenden Scans der gegebenen Anwendungskomponente nach Ergreifen einer Behebungsmaßnahme bei der gegebenen Anwendungskomponente, basierend auf einer Bestimmung, dass sich eine Bedrohung, die mit der Schwachstelle verknüpft ist, geändert hat;
Bestimmen, basierend auf dem folgenden Scan der gegebenen Anwendungskomponente, dass die Behebungsmaßnahme mindestens einer der Schwachstellen behoben hat, die mit der gegebenen Anwendungskomponente verknüpft sind, die vor Ergreifen der Behebungsmaßnahme an der gegebenen Anwendungskomponente vorhanden waren;
Bestimmen, anhand des folgenden Scans der gegebenen Anwendungskomponente, einer oder mehrerer neuer Schwachstellen, die mit der gegebenen Anwendungskomponente verknüpft sind;
Bestimmen, aus einem Zwischenspiel der Schwachstellen der gegebenen Anwendungskomponente, die die eine oder mehreren neuen Schwachstellen beinhaltet, dass eine Korrekturmaßnahme erforderlich ist; und
Einstellen einer Firewall-Vorrichtung, um mindestens eine der einen oder mehreren neuen Schwachstellen für die Korrekturmaßnahme zu bekämpfen, wobei die Firewall-Vorrichtung verwendet wird, um mehrere Sicherheitszonen zu errichten.

9. Verfahren nach Anspruch 8, weiter beinhaltend:

Bestimmen eines Ausmaßes der Schwachstellen; und Reihen der Schwachstellen; oder
Bestimmen eines Ausmaßes der Schwachstellen, beinhaltend einen zusammengesetzten Punktewert aus einer Vielzahl von Schwachstellenfaktoren; und Reihen der Schwachstellen.

10. Verfahren nach Anspruch 8, weiter umfassend Identifizieren von Pentest-Vorschlägen, um weiter ein Vorhandensein von Schwachstellen basierend auf der Identität der Schwachstellen zu bestätigen.

11. Verfahren nach Anspruch 8, weiter umfassend Ergreifen einer Behebungsmaßnahme in der Form geänderter Eindringerfassungs- oder Schutzregeln, basierend auf der Identität der Schwachstellen.

12. Verfahren nach Anspruch 8, wobei Einstellen der Charakterisierungen der Schwachstellen weiter basiert auf
einer Sicherheitskonfiguration des Systems, die sich auf die Schwachstellen bezieht; oder
Kontextinformationen, die Bedrohungsvektoren, Bedrohungseinflüsse und Unternehmensreife des Systems beinhalten; oder
mindestens auf zwei oder mehr von einem Zugriffsvektor, Zugriffskomplexität, Authentifizierung, Vertrauenseinfluss, Integritätseinfluss, Verfügbarkeitseinfluss, Nutzbarkeit, Behebungsniveau, Berichtsicherheit, Kollateralschadenpotential, Zielverteilung, Vertraulichkeit, Integritätsanforderung und Verfügbarkeitsanforderung oder
einer bestimmten Änderung in einer Bedrohung, die die Schwachstelle verursacht.

13. Verfahren nach Anspruch 8, wobei die eine oder mehreren neuen Schwachstellen mehrere neue Schwachstellen beinhalten.

14. Verfahren nach Anspruch 8, wobei Einstellen der Netzwerksicherheitsvorrichtung zum Bekämpfen der mindestens einen der einen oder mehreren neuen Schwachstellen für die Behebungsmaßnahme Einstellen der Netzwerksicherheitsvorrichtung beinhaltet, um eine erste neue Schwachstelle der neuen Schwachstellen zu bekämpfen und eine zweite neue Schwachstelle der neuen Schwachstellen wegzulassen.

15. Mindestens ein maschinenlesbares Speichermedium, das computerausführbare Anweisungen umfasst, die auf dem maschinenlesbaren Speichermedium getragen werden, um den Prozessor zu veranlassen, das Verfahren nach einem der Ansprüche 8 bis 14 durchzuführen

Revendications

1. Système de sécurité électronique, comprenant :

un processeur ; et
une mémoire couplée de manière communicative au processeur et incluant des instructions, les instructions, lorsqu'elles sont chargées et exécutées par le processeur, amènent le processeur à :

analyser, lors d'un premier balayage, une structure de fichier d'application d'une application et un tableau d'importation de l'application pour identifier, via la structure de fichier d'application de l'application et le tableau d'importation de l'application, un ou plusieurs composants d'application identifiés de manière unique ;
déterminer des vulnérabilités associées à un composant d'application donné des un ou plusieurs composants d'application identifiés de manière unique, les vulnérabilités incluant les vulnérabilités d'un ou de plusieurs composants supplémentaires auxquels le composant d'application donné doit accéder ;
ajuster les caractérisations des vulnérabilités associées au composant d'application donné sur la base d'informations contextuelles provenant du système dans lequel réside le composant d'application donné, les informations contextuelles incluant des informations de sécurité ; et
réaliser une action corrective sur la base des caractérisations de vulnérabilité ajustées ; et
après réalisation de l'action corrective sur le composant d'application donné :

effectuer un balayage ultérieur du composant d'application donné après réalisation d'une action corrective sur le composant d'application donné, sur la base d'une détermination du fait qu'une menace associée à la vulnérabilité a changé ;
déterminer, sur la base du balayage ultérieur du composant d'application donné, que l'action corrective a corrigé au moins une des vulnérabilités associées au composant d'application donné, présent avant réalisation de l'action corrective sur le composant d'application donné ;
déterminer, sur la base du balayage ultérieur du composant d'application donné, une ou plusieurs nouvelles vulnérabilités associées au composant d'application donné ;
déterminer à partir d'une interaction des vulnérabilités du composant d'application donné, qui inclut les une ou plusieurs nouvelles vulnérabilités, qu'une action corrective est nécessaire ; et
ajuster un dispositif pare-feu pour vaincre au moins une des une ou plusieurs nouvelles vulnérabilités pour l'action corrective, dans lequel le dispositif pare-feu est utilisé pour établir de multiples zones de sécurité.

2. Système selon la revendication 1, dans lequel la mémoire inclut en outre des instructions pour amener le processeur à :

déterminer une mesure des vulnérabilités, et classer les vulnérabilités ; ou
déterminer une mesure des vulnérabilités incluant un score composite à partir d'une pluralité de facteurs de vulnérabilité, et classer les vulnérabilités.

3. Système selon la revendication 1, dans lequel la mémoire inclut en outre des instructions pour amener le processeur à identifier des suggestions de test d'intrusion pour confirmer en outre la présence de vulnérabilités sur la base de l'identité des vulnérabilités.

4. Système selon la revendication 1, dans lequel, pour réaliser une action corrective, la mémoire inclut en outre des instructions pour amener le processeur à réaliser l'action corrective sous la forme de règles de détection d'intrusion ou de protection changées sur la base de l'identité des vulnérabilités.

5. Système selon la revendication 1, dans lequel l'ajustement des caractérisations des vulnérabilités est en outre basé sur :

une configuration de sécurité du système relative aux vulnérabilités ; ou
des informations contextuelles incluant des vecteurs de menace, des impacts de menace et une maturité d'entreprise du système ; ou
au moins sur deux ou plus d'un vecteur d'accès, d'une complexité d'accès, d'une authentification, d'un impact confidentiel, d'un impact d'intégrité, d'un impact de disponibilité, d'une exploitabilité, d'un niveau de remédiation, d'une confiance de rapport, d'un potentiel de dommage collatéral, d'une distribution cible, d'une confidentialité, d'une exigence d'intégrité, d'une exigence de disponibilité, ou

un changement déterminé d'une menace causant la vulnérabilité.

6. Système selon la revendication 1, dans lequel les une ou plusieurs nouvelles vulnérabilités incluent de multiples nouvelles vulnérabilités.

7. Système selon la revendication 1, dans lequel l'ajustement du dispositif de sécurité de réseau pour vaincre l'au moins une des une ou plusieurs nouvelles vulnérabilités pour l'action corrective inclut l'ajustement du dispositif de sécurité de réseau pour vaincre une première nouvelle vulnérabilité des nouvelles vulnérabilités et omettre une seconde nouvelle vulnérabilité des nouvelles vulnérabilités.

8. Procédé de sécurité électronique, comprenant
l'analyse, lors d'un premier balayage, d'une structure de fichier d'application d'une application et d'un tableau d'importation de l'application pour identifier, via la structure de fichier d'application de l'application et le tableau d'importation de l'application, un ou plusieurs composants d'application identifiés de manière unique ;
le balayage des un ou plusieurs composants d'application identifiés de manière unique pour déterminer des vulnérabilités associées à un composant d'application donné des un ou plusieurs composants d'application identifiés de manière unique, les vulnérabilités incluant les vulnérabilités d'un ou de plusieurs composants supplémentaires auxquels le composant d'application donné doit accéder ;
l'ajustement de caractérisations des vulnérabilités associées au composant d'application donné sur la base au moins d'informations contextuelles provenant du système dans lequel réside le composant d'application donné, les informations contextuelles incluant des informations de sécurité ; et
la réalisation d'une action corrective sur la base des caractérisations de vulnérabilité ajustées ; et
après réalisation de l'action corrective sur le composant d'application donné :

la réalisation d'un balayage ultérieur du composant d'application donné après réalisation d'une action corrective sur le composant d'application donné, sur la base d'une détermination du fait qu'une menace associée à la vulnérabilité a changé ;
la détermination, sur la base du balayage ultérieur du composant d'application donné, du fait que l'action corrective a corrigé au moins une des vulnérabilités associées au composant d'application donné, présent avant réalisation de l'action corrective sur le composant d'application donné ;
la détermination, lors du balayage ultérieur du composant d'application donné, d'une ou de plusieurs nouvelles vulnérabilités associées au composant d'application donné ;
la détermination à partir d'une interaction des vulnérabilités du composant d'application donné, qui inclut les une ou plusieurs nouvelles vulnérabilités, du fait qu'une action corrective est nécessaire ; et
l'ajustement d'un dispositif pare-feu pour vaincre au moins une des une ou plusieurs vulnérabilités pour l'action corrective, dans lequel le dispositif pare-feu est utilisé pour établir de multiples zones de sécurité.

9. Procédé selon la revendication 8, incluant en outre :

la détermination d'une mesure des vulnérabilités, et le classement des vulnérabilités ; ou
la détermination d'une mesure des vulnérabilités incluant un score composite à partir d'une pluralité de facteurs de vulnérabilité ; et le classement des vulnérabilités.

10. Procédé selon la revendication 8, comprenant en outre l'identification de suggestions de test d'intrusion pour confirmer une présence des vulnérabilités sur la base de l'identité des vulnérabilités.

11. Procédé selon la revendication 8, comprenant en outre la réalisation d'une action corrective sous la forme de règles de détection d'intrusion ou de protection changées sur la base de l'identité des vulnérabilités.

12. Procédé selon la revendication 8, dans lequel l'ajustement des caractérisations des vulnérabilités est en outre basé sur :

une configuration de sécurité du système relative aux vulnérabilités ; ou
des informations contextuelles incluant des vecteurs de menace, des impacts de menace et une maturité d'entreprise du système ; ou
au moins sur deux ou plus d'un vecteur d'accès, d'une complexité d'accès, d'une authentification, d'un impact confidentiel, d'un impact d'intégrité, d'un impact de disponibilité, d'une exploitabilité, d'un niveau de remédiation, d'une confiance de rapport, d'un potentiel de dommage collatéral, d'une distribution cible, d'une confidentialité, d'une exigence d'intégrité, d'une exigence de disponibilité, ou
un changement déterminé d'une menace causant la vulnérabilité.

13. Procédé selon la revendication 8, dans lequel les une ou plusieurs nouvelles vulnérabilités incluent de multiples nouvelles vulnérabilités.

14. Procédé selon la revendication 8, dans lequel l'ajustement du dispositif de sécurité de réseau pour vaincre l'au moins une des une ou plusieurs nouvelles vulnérabilités pour l'action corrective inclut l'ajustement du dispositif de sécurité de réseau pour vaincre une première nouvelle vulnérabilité des nouvelles vulnérabilités et omettre une seconde vulnérabilité des nouvelles vulnérabilités.

15. Au moins un support de stockage lisible par machine, comprenant des instructions exécutables par ordinateur portées par le support de stockage lisible par machine, les instructions étant lisibles par un processeur, les instructions, lorsqu'elles sont lues et exécutées, permettant d'amener le processeur à réaliser le procédé selon l'une quelconque des revendications 8 à 14.

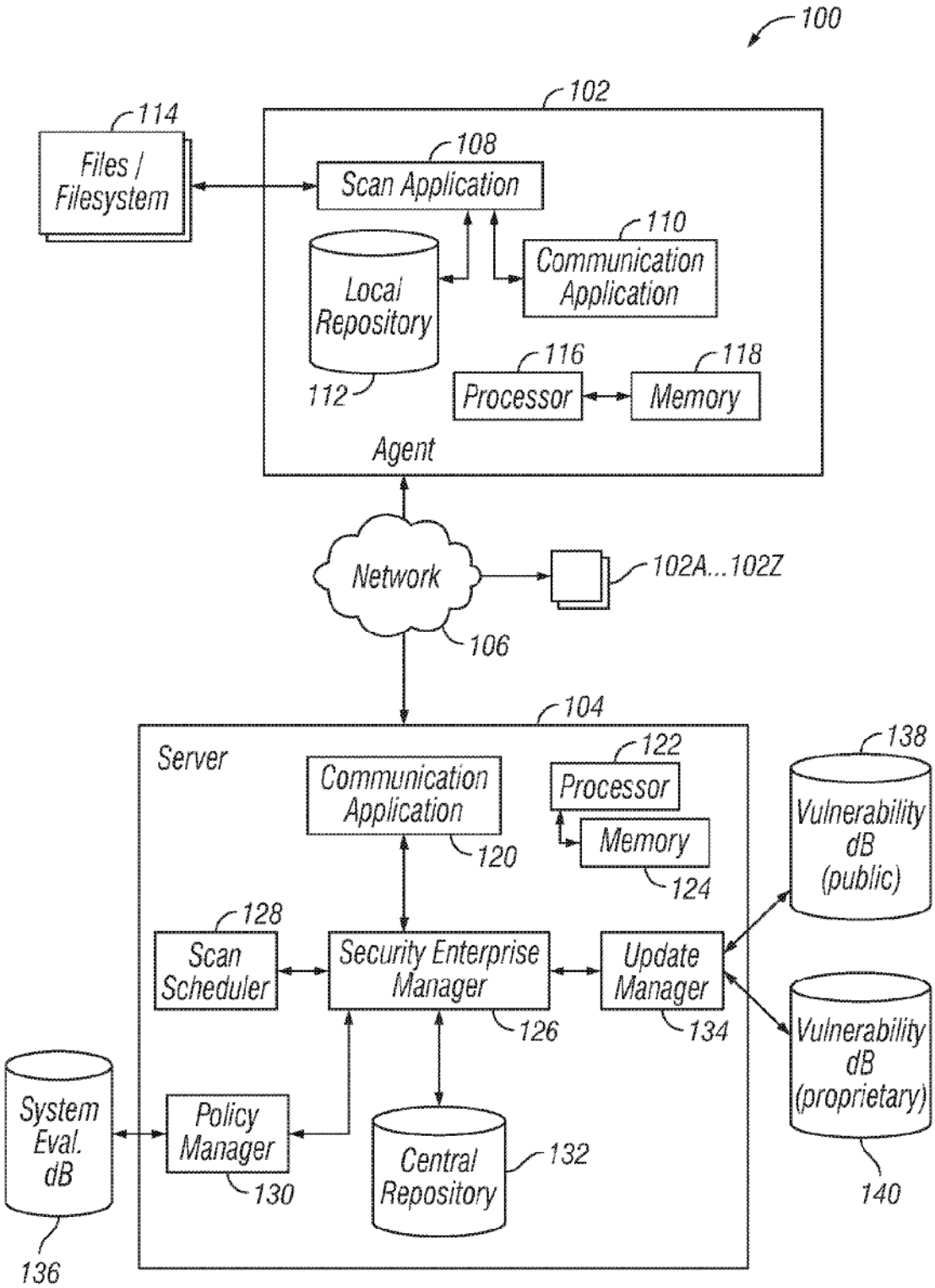
100
102
114
Files / Filesystem
108
Scan Application
110
Communication Application
Local Repository
112
116
Processor
118
Memory
Agent
Network
102A...102Z
106
104
Server
Communication Application
120
122
Processor
Memory
124
138
Vulnerability dB (public)
128
Scan Scheduler
Security Enterprise Manager
126
Update Manager
134
Vulnerability dB (proprietary)
140
System Eval. dB
136
Policy Manager
130
Central Repository
132

FIG. 1

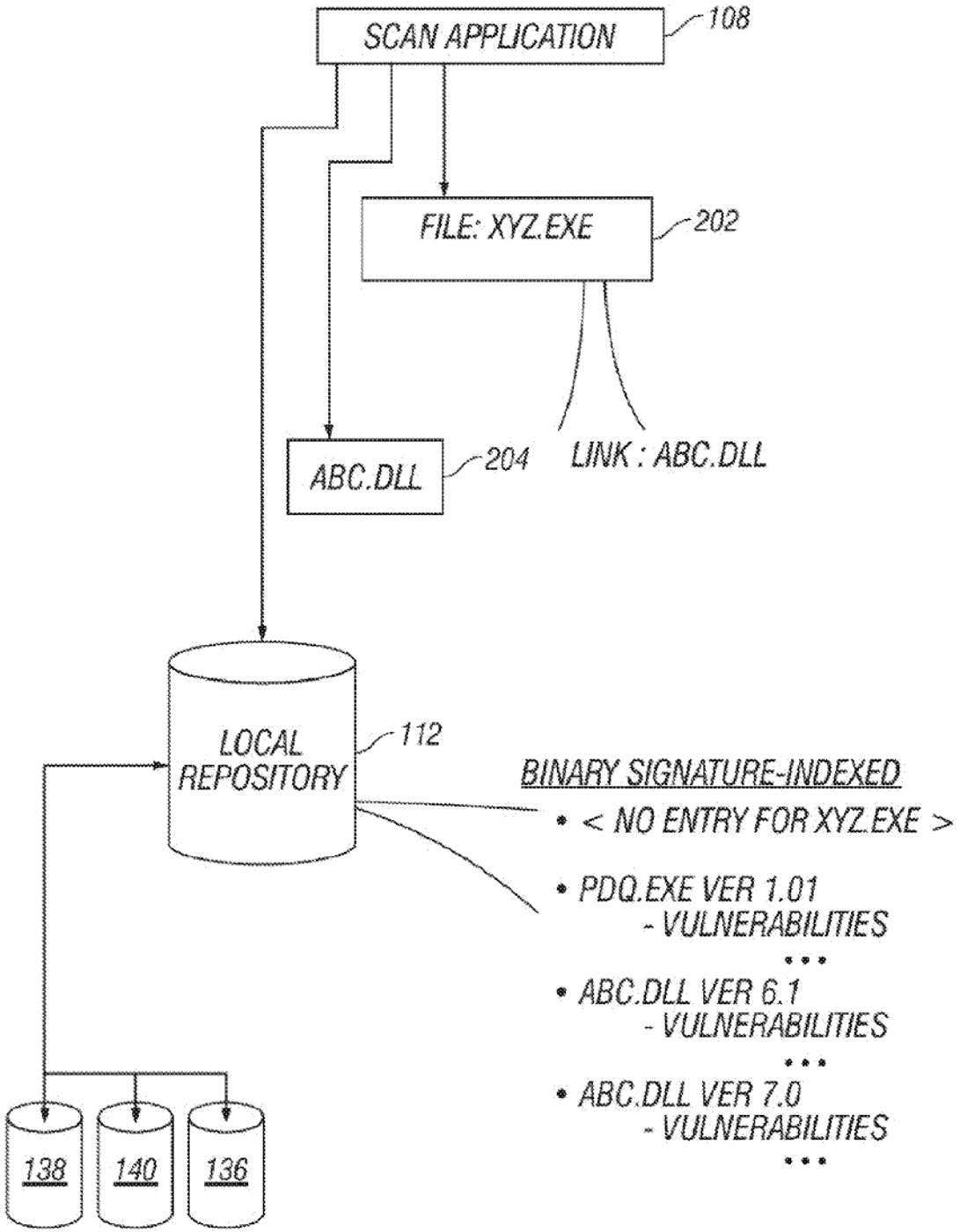
SCAN APPLICATION
108
FILE: XYZ.EXE
202
ABC.DLL
204
LINK : ABC.DLL
LOCAL REPOSITORY
112
BINARY SIGNATURE-INDEXED
• < NO ENTRY FOR XYZ.EXE >
• PDQ.EXE VER 1.01
- VULNERABILITIES
• • •
• ABC.DLL VER 6.1
- VULNERABILITIES
• • •
• ABC.DLL VER 7.0
- VULNERABILITIES
• • •
138
140
136

FIG. 2

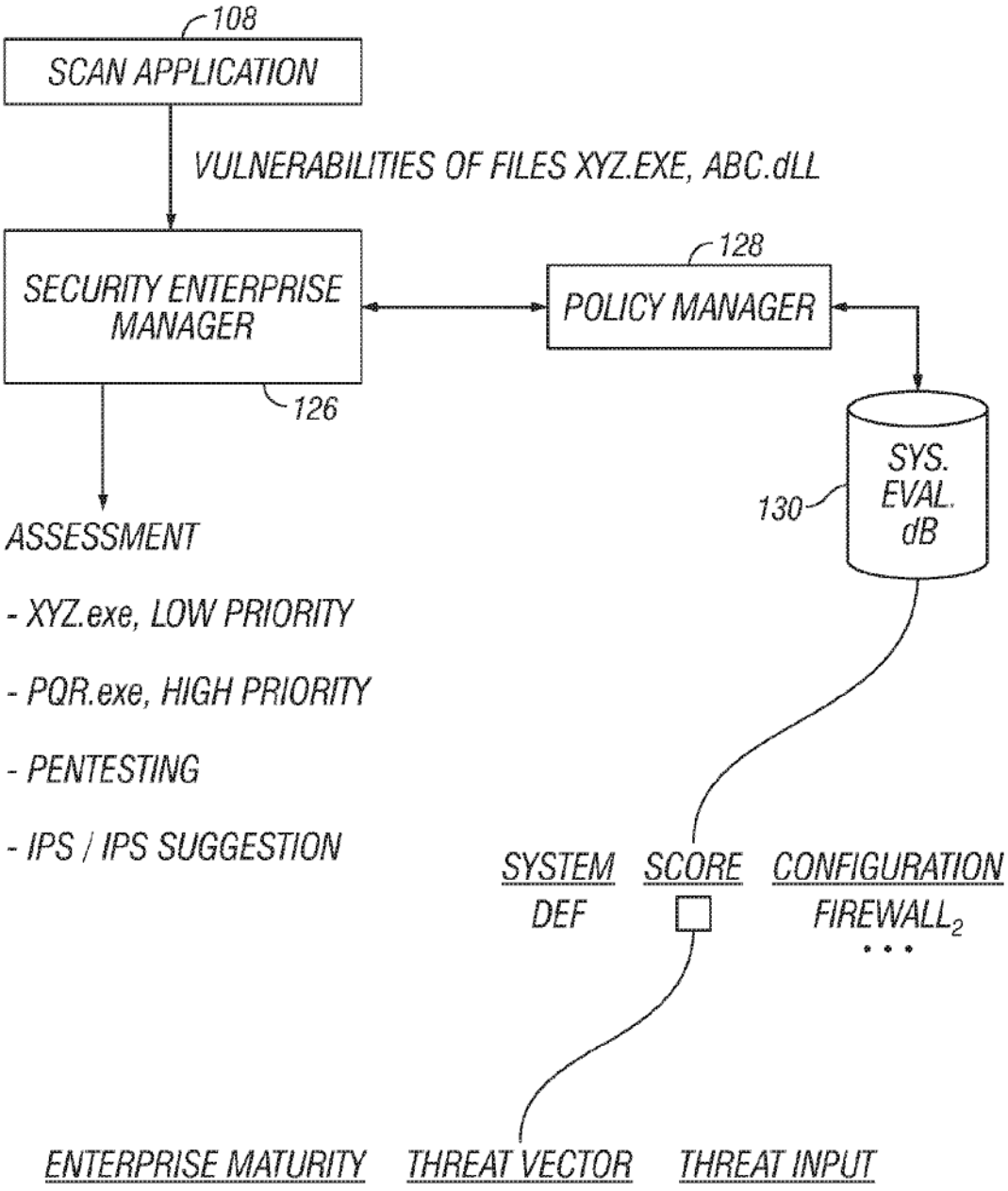
108
SCAN APPLICATION
VULNERABILITIES OF FILES XYZ.EXE, ABC.dLL
SECURITY ENTERPRISE MANAGER
126
128
POLICY MANAGER
SYS. EVAL. dB
130
ASSESSMENT
- XYZ.exe, LOW PRIORITY
- PQR.exe, HIGH PRIORITY
- PENTESTING
- IPS / IPS SUGGESTION
SYSTEM DEF
SCORE
CONFIGURATION FIREWALL2
ENTERPRISE MATURITY
THREAT VECTOR
THREAT INPUT

FIG. 3

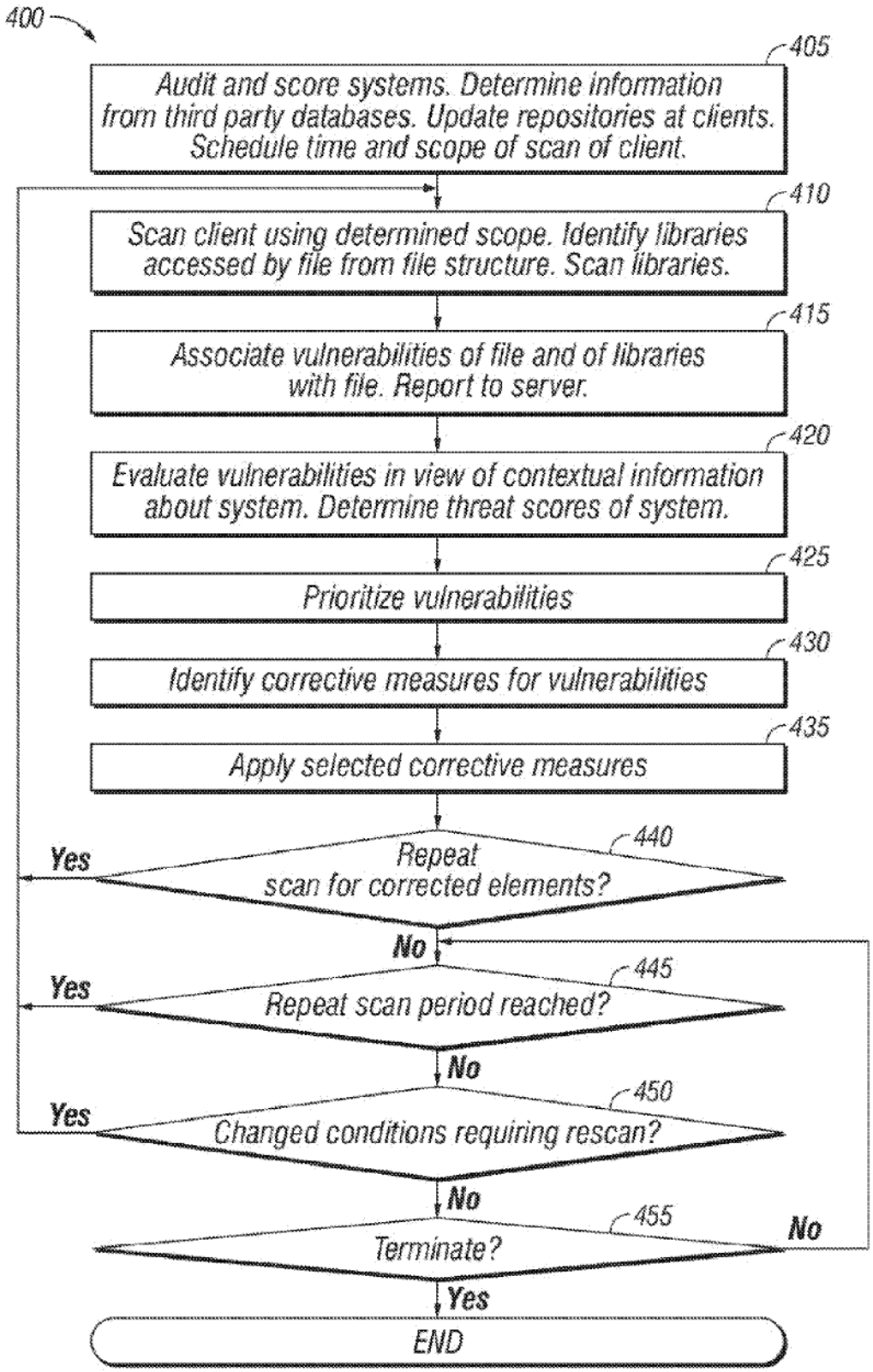
400
405
Audit and score systems. Determine information from third party databases. Update repositories at clients. Schedule time and scope of scan of client.
410
Scan client using determined scope. Identify libraries accessed by file from file structure. Scan libraries.
415
Associate vulnerabilities of file and of libraries with file. Report to server.
420
Evaluate vulnerabilities in view of contextual information about system. Determine threat scores of system.
425
Prioritize vulnerabilities
430
Identify corrective measures for vulnerabilities
435
Apply selected corrective measures
440
Repeat scan for corrected elements?
Yes
No
445
Repeat scan period reached?
Yes
No
450
Changed conditions requiring rescan?
Yes
No
455
Terminate?
No
Yes
END

FIG. 4

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 2011068967 A **[0005]**