(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	

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 associated 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

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.