Title: AUGMENTING SYSTEM RESTORE WITH MALWARE DETECTION

Abstract: An anti-malware program monitors the behavior of a system after a system restore to determine the likelihood of a hidden infection of malicious code still existing after the system restore. The anti-malware program observes the dynamic behavior of the system by monitoring conditions that are likely to signify the possibility of an infection thereby necessitating the need to initiate anti-malware detection. The anti-malware program may observe the restoration history, system settings, malware infection history, to determine the likelihood of an existing hidden infection after a system restore.

FIG. 1

(a) International Application Number: PCT/US2012/069978
(b) International Filing Date: 17 December 2012 (17.12.2012)

(a) Priority Data: 13/334,060 22 December 2011 (22.12.2011) US

(a) International Classification: G06F 21/56 (2013.01) G06F 15/16 (2006.01)

Declarations under Rule 4.17:
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(H))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(in))

Published:
- with international search report (Art. 21(3))
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))

Title: AUGMENTING SYSTEM RESTORE WITH MALWARE DETECTION

Abstract: An anti-malware program monitors the behavior of a system after a system restore to determine the likelihood of a hidden infection of malicious code still existing after the system restore. The anti-malware program observes the dynamic behavior of the system by monitoring conditions that are likely to signify the possibility of an infection thereby necessitating the need to initiate anti-malware detection. The anti-malware program may observe the restoration history, system settings, malware infection history, to determine the likelihood of an existing hidden infection after a system restore.
AUGMENTING SYSTEM RESTORE WITH MALWARE DETECTION

BACKGROUND

A computer anti-malware program aims to detect, prevent, and remove malicious code from executing on a computing device and creating unwanted consequences. Malware may take the form of a computer virus, computer worm, spyware, adware, a Trojan horse, a rootkit, and so forth. Malware or malicious code may add, delete, or tamper with segments of programs, files, memory locations, or boot sectors. An anti-malware program may be utilized to scan storage devices for files and memory locations that may have been infected and corrupted. The anti-malware program may scan a file when it is installed onto a computer system or whenever the file is opened. The anti-malware program may also be scheduled to scan files or memory locations at regularly scheduled intervals. When malware is detected, remediation may be performed to repair a corrupted portion of a file, to restore the corrupted file to a clean state, or to remove the infected portion from a file. However, the remedial measures may not be sufficient since the source of the infection may still be present after the remediation.

SUMMARY

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

Anti-malware detection is performed after a system restoration that restores an operating system in a computing device in order to ensure the integrity of a system from a hidden infection that may be residing in user files. An anti-malware program monitors the dynamic behavior of a system for conditions that signify the possibility of an infection thereby necessitating the need to initiate malware detection. The indicators utilize conditions exhibited during the operation of the system that are more likely to be predictive of an infection. Such conditions may include the restoration history, the infection history of the system, the detection of disabled security settings, and so forth. Heuristic rules are applied to determine whether or not malware detection is warranted when a condition occurs and the level of detection that may be needed.

These and other features and advantages will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be
understood that both the foregoing general description and the following detailed description are explanatory only and are not restrictive of aspects as claimed.

**BRIEF DESCRIPTION OF DRAWINGS**

[0006] Fig. 1 illustrates an exemplary system that augments a system restore with malware detection.

[0007] Fig. 2 is a flow diagram illustrating a first exemplary method.

[0008] Fig. 3 is a flow diagram illustrating a second exemplary method.

[0009] Fig. 4 is a flow diagram illustrating a third exemplary method.

[0010] Fig. 5 is a flow diagram illustrating a fourth exemplary method.

[0011] Fig. 6 is a flow diagram illustrating a fifth exemplary method.

[0012] Fig. 7 is a flow diagram illustrating a sixth exemplary method.

[0013] Fig. 8 is a block diagram illustrating an operating environment.

[0014] Fig. 9 is a block diagram illustrating an exemplary computing device.

**DETAILED DESCRIPTION**

[0015] Various embodiments are directed to a technology that augments a system restore with anti-malware detection in order to ensure the integrity of a system. A system restore is often performed by a user or system administrator when a widespread malware infection is highly probable and restores an operating system to a previous state. The operating system may be restored from an installation media or from a partition in a storage device. However, the user files, which may not have been restored, may be infected with malware and present a risk for re-infecting the operating system files.

[0016] An anti-malware program may observe the dynamic behavior of a system for conditions that signify the possibility of an infection thereby necessitating the need to initiate malware detection. The indicators utilize conditions exhibited during the operation of the system and heuristic rules that are more likely to be predictive of an infection.

Attention now turns to a more detailed description of such a system.

[0017] Fig. 1 illustrates a block diagram of an exemplary system 100 that may include a computing device 102 and a server 104 communicatively coupled through a network 106. Although the system 100 as shown in Fig. 1 has a limited number of elements in a certain topology, it may be appreciated that the system 100 may include more or less elements in alternate topologies as desired for a given implementation.

[0018] The computing device 102 and the server 104 may be any type of electronic device capable of executing programmable instructions such as, without limitation, a mobile device, a personal digital assistant, a mobile computing device, a smart phone, a
cellular telephone, a handheld computer, a server, a server array or server farm, a web
server, a network server, an Internet server, a work station, a mini-computer, a mainframe
computer, a supercomputer, a network appliance, a web appliance, a distributed computing
system, multiprocessor systems, or combination thereof. The network 106 may be any
type of communications link capable of facilitating communications between the
computing device 102 and the server 104, utilizing any communications protocol and in
any configuration, such as without limitation, a wired network, wireless network, or
combination thereof. It will be appreciated that the network connections shown are
exemplary and other means of establishing a communications link between the computing
device 102 and the server 104 may be used as well.

The computing device 102 may include an anti-malware program 108 that
monitors, detects, and attempts to prevent malicious code from infecting resources on the
computing device 102. A resource may be a file, program, registry, operating system
configurations and settings, and so forth. Malware or malicious code may attempt to
prevent the anti-malware program to add, delete, or tamper with segments of resources
and/or storage areas and cause unauthorized and unwanted consequences. Malware may
be a computer virus, computer worm, spyware, adware, a Trojan horse, a rootkit, and the
like. The anti-malware program 108 may also remediate an infection by restoring the
infected resource with a clean version, remove the infection from a resource, or quarantine
the resource. The anti-malware program 108 may be part of an operating system, security
application, or other software application. In addition, the anti-malware program 108 may
be a stand-alone executable program that may reside in the computing device or execute
from a remote server through a web browser. The embodiments are not limited in this
manner.

The anti-malware program 108 may be a sequence of computer program
instructions, that when executed by a processor, causes the processor to perform methods
and/or operations in accordance with a prescribed task. The anti-malware program 108
may be implemented as program code, programs, procedures, module, code segments,
program stacks, middleware, firmware, methods, routines, and so on. The executable
computer program instructions may be implemented according to a predefined computer
language, manner or syntax, for instructing a computer to perform a certain function. The
instructions may be implemented using any suitable high-level, low-level, object-oriented,
visual, compiled and/or interpreted programming language.
The computing device 102 may include an operating system 110 that manages and controls the system resources, such as the hardware devices connected to the computing device 102, and the software applications that execute therein. The operating system 110 may store system settings 112 that are used to control certain features provided by the operating system 110. The system settings 112 may control performance options (e.g., processor scheduling, virtual memory paging file size, memory usage, etc.), display options (e.g., type of wallpaper, location of icons, display of windows, colors, sound, etc.), network settings (e.g., browser options, bookmarks, home page URL, etc.), email options (e.g., mail rules, views in the mailbox, mail server, etc.), and the like.

The system settings 112 may include security settings that protect the computer system from unwanted actions and consequences. For example, the security settings may include operating system update settings, firewall settings, browser settings, malware dictionary update settings, user access control settings, and so forth. An operating system update setting, when enabled, allows the computing device 102 to receive updates to the operating system, such as security updates, vulnerability fixes, bug fixes, and so on. A firewall setting enables a firewall to monitor network transmissions in order to prevent unauthorized access. The browser settings allow the browser to analyze downloaded web pages for suspicious characteristics, check for well known malicious web sites, check downloaded files for malicious software, and so forth. The malware dictionary settings allow the computing device 102 to control the manner in which updates are made to a malware dictionary. The user access control settings prevent unauthorized access to crucial settings that when modified affect the operation of the computing device.

The operating system 110 may also include an infection history 114 that tracks previous malware infections within the computing device 102. The infection history 114 may indicate the day and time an infection was detected, the physical location of the infection, the type of infection, and so forth.

A restore program 116 may be a software application that restores the operating system 110 or files to a previous uninfected or clean state. The restore program 116 may restore the operating system files from an installation media (e.g., CD, DVD, flash drive, etc.), a partition of a hard drive, or a remotely located storage device. The restore program 116 may restore the resources or storage areas back to a restore point known not to have an infection. A restore point may be generated upon the occurrence of an important event or periodically in accordance with a schedule. For example, the restore program 116 may monitor resources for modifications and store a copy of the resource or storage area before...
the modifications are made. The restore program 116 may also be configured to record and store certain storage areas and/or resources as needed. In the event of a problem, the restore program 116 may restore the resource and/or storage area back to the restore point chosen by the user.

5 [0025] A restore counter 118 may be utilized to determine the number of times that a system restoration has occurred within a particular time frame. A system restoration may be initiated by a user or system administrator upon the occurrence of a catastrophic event or by the user. The restore program 116 will update the restore counter 118 each time a system restore operation is performed.

10 [0026] The computing device 102 may also include a malware dictionary 122 that includes signatures 124 used to identify malware. A signature 124 may be a hash-based code or other type of coded representation that identifies a malware program. Variants of a well known malware program may be created that have a similar behavior to a well known malware program with a different signature. Heuristic rules may be used to determine the signature 124 of a malware program. The malware dictionary 122 is continuously updated to include the signatures of newly detected malware programs and their variants. The updates 126 may be provided by a server 104 hosting a trusted service that specializes in aggregating data pertaining to malware programs from various sources or from other computing systems connected to the computing device 102.

15 [0027] In several embodiments, the server 104 may be hosting an application store that provides the computing device 102 with software applications downloaded through a licensing arrangement. The application store may provide the computing device with updates 126 of previously found malware programs that may have been contained in a software application downloaded to the computing device 102. The updates 126 may contain signatures of known malware programs that are stored in the malware dictionary 122.

[0028] The anti-malware program 108 may receive data from various components of the computing device 102 in order to analyze the behavior of the system. The anti-malware program 108 may utilize the restore counter 118, the system settings 112, and the infection history 114 as indicators of the dynamic behavior of the system which are applied against a set of heuristic rules. The result of the analysis is to predict the risk level of an infection and the need to scan the storage areas according to the predicted risk level.

[0029] Upon determining the threat of a risk and the associated risk level, the anti-malware program may initiate a scan operation 128 to a storage area 130. The storage
area 130 may be used to store files and data used in the computing device 102. The
storage area 130 may be implemented as a semiconductor memory device, a computer-
readable media, a hard disk drive, any combination thereof, or as any other type of device
capable of storing data.

In various embodiments, the system 100 described herein may comprise a
computer-implemented system having multiple elements, programs, procedures, modules.
As used herein, these terms are intended to refer to a computer-related entity, comprising
either hardware, a combination of hardware and software, or software. For example, an
element may be implemented as a process running on a processor, a hard disk drive,
multiple storage drives (of optical and/or magnetic storage medium), an object, an
executable, a thread of execution, a program, and/or a computer. By way of illustration,
both an application running on a server and the server may be an element. One or more
elements may reside within a process and/or thread of execution, and an element may be
localized on one computer and/or distributed between two or more computers as desired
for a given implementation. The embodiments are not limited in this manner.

The various elements of system 100 may be communicatively coupled via
various types of communications medium as indicated by various lines or arrows. The
elements may coordinate operations between each other. The coordination may involve
the uni-directional or bi-directional exchange of information. For instance, the elements
may communicate information in the form of signals communicated over the
communications medium. The information may be implemented as signals allocated to
various signal lines. In such allocations, each message is a signal. Further embodiments,
however, may alternatively employ data messages. Such data messages may be sent
various connections. Exemplary connections include parallel interfaces, serial interfaces,
and bus interfaces.

Attention now turns to a discussion of the operations for the embodiments with
reference to various exemplary methods. It may be appreciated that the representative
methods do not necessarily have to be executed in the order presented, or in any particular
order, unless otherwise indicated. Moreover, various activities described with respect to
the methods can be executed in serial or parallel fashion, or any combination of serial and
parallel operations. The methods can be implemented using one or more hardware
elements and/or software elements of the described embodiments or alternative
embodiments as desired for a given set of design and performance constraints. For
example, the methods may be implemented as logic (e.g., computer program instructions) for execution by a logic device (e.g., a general-purpose or specific-purpose computer).

[0033] Fig. 2 illustrates a flow diagram of an exemplary method 200 for augmenting a system restore with malware detection. It should be noted that the method 200 may be representative of some or all of the operations executed by one or more embodiments described herein and that the method can include more or less operations than that which is described in Fig. 2.

[0034] A system restore operation may be initiated to restore the operating system 110 files (block 202). The anti-malware program 108 may then be initiated to determine the level of scanning desired based on the behavior of the system and to scan at the highest level determined (block 204). The anti-malware program 108 may execute during the system restoration or after the system restoration completes. The level of scanning may include a deep scan level and/or a user file scan level. A deep scan level scans every file and all memory locations in the storage area 130. A user file scan level scans the user files, operating system files, and all memory locations in the storage area 130. The deep scan level is a higher level of scanning than the user file scan level and as such, takes precedence over the user files scan level.

[0035] If the anti-malware program 108 detects an infected file or area of the storage area 130, then an appropriate remediation measure is taken (block 206). The remediation measure may restore a file or area of the storage area 130, remove the infected file or portion thereof, quarantine the file, and so forth (block 206).

[0036] Fig. 3 is a flow diagram illustrating an exemplary method 204 for determining whether scanning may be needed. It should be noted that the method 204 may be representative of some or all of the operations executed by one or more embodiments described herein and in alternate configurations. The method can include more or less operations than that which is described in Fig. 3 and the order in which each check is made may be performed differently than the order shown in Fig. 3. Furthermore, the checks performed in each block shown in Fig. 3 may be performed in parallel with other checks. The embodiments are not constrained in this manner.

[0037] The anti-malware program 108 may check the value of the restore counter 118 to determine the level of scanning that may be needed (block 220). The restore counter 118 may be used to reflect the restoration history of the computing device 102. Referring to Fig. 4, in the event the restore counter 118 exceeds a threshold (block 230-yes), then a deep scan level may be determined and set (block 232). The threshold may be a
configurable value that signifies a potential risk of infection based on too many restorations having been performed. If the threshold has not been met (block 230-no), then a user file level scan is determined and set (block 234). The scan of the user files may be needed in the event the restoration restored only system-level files, such as the operating system files, and not user files where the source of an infection may be hidden.

Referring back to Fig. 3, the anti-malware program 108 may check certain system settings to determine whether or not scanning is needed and if so, the level scanning needed (block 222). Referring to Fig. 5, the anti-malware program 108 may analyze certain system settings which may be classified as critical and non-critical (block 240). In several embodiments, the security settings, such as the operating system update settings, firewall settings, browser settings, malware dictionary update settings, and user access control settings, may be deemed critical settings. If all of the critical settings are not enabled (block 242-no), then the anti-malware program 108 determines that a deep scan level is warranted (block 246). Otherwise (block 242-yes), if the number of non-critical settings that are disabled exceeds a threshold amount (block 244-yes), then a deep scan level is warranted (block 246). Otherwise, when all critical and non-critical settings are enabled (block 244-no), the process returns.

Referring back to Fig. 3, the anti-malware program 108 may check the infection history 114 to determine whether scanning is needed and if so, the level of scanning needed (block 224). The infection history 114 may be stored in a memory space associated with the operating system 110 and may also be stored in a memory space associated with the anti-malware program 108. Referring to Fig. 6, the anti-malware program 108 may analyze the infection history 114 to determine whether the time since the last infection exceeds a threshold (block 250-yes). If the threshold is exceeded (block 250-yes), then a deep scan level is set (block 252). Otherwise when the threshold is not exceeded (block 250-no), then the process returns.

Referring back to Fig. 3, the anti-malware program 108 may check with a server 104 hosting an application store for known malware programs that are associated with applications that were previously obtained from the application store and downloaded to the computing device 102 (block 226). Referring to Fig. 7, the anti-malware program 108 may query the application store to find out if any recently downloaded applications are known to contain malware or otherwise produce unwanted consequences (block 260). In the event a recently downloaded application is known to contain malware (block 262-yes), then the anti-malware program 108 may determine a deep level scan is warranted (block
264). Otherwise, if the application store is not aware of malware contained in recently
downloaded applications (block 262-no), then the process returns.

[0041] Attention now turns to a discussion of an exemplary operating environment. Fig.
8 illustrates an operating environment 300. It should be noted that the operating
environment 300 is exemplary and is not intended to suggest any limitation as to the
functionality of the embodiments. The embodiment may be applied to an operating
environment 300 having one or more client(s) 302 in communication through a
communications framework 304 with one or more server(s) 306. The operating
environment 300 may be configured in a network environment, a distributed environment,
a multiprocessor environment, or a stand-alone computing device having access to remote
or local storage devices.

[0042] A client 302 may be embodied as a hardware device, a software module, or as a
combination thereof. Examples of such hardware devices may include, but are not limited
to, a computer (e.g., server, personal computer, laptop, etc.), a cell phone, a personal
digital assistant, or any type of computing device, and the like. A client 302 may also be
embodied as a software module having instructions that execute in a single execution path,
multiple concurrent execution paths (e.g., thread, process, etc.), or in any other manner.

[0043] A server 306 may be embodied as a hardware device, a software module, or as a
combination thereof. Examples of such hardware devices may include, but are not limited
to, a computer (e.g., server, personal computer, laptop, etc.), a cell phone, a personal
digital assistant, or any type of computing device, and the like. A server 306 may also be
embodied as a software module having instructions that execute in a single execution path,
multiple concurrent execution paths (e.g., thread, process, etc.), or in any other manner.

[0044] The communications framework 304 facilitates communications between the
client 302 and the server 306. The communications framework 304 may embody any type
of communications medium, such as wired or wireless networks, utilizing any
communication protocol. Each client(s) 302 may be coupled to one or more client data
store(s) 308 that store information local to the client 302. Each server(s) 306 may be
coupled to one or more server data store(s) 310 that store information local to the server
306.

[0045] Fig. 9 illustrates a block diagram of an exemplary computing device 102. The
computing device 102 may have one or more processors 404, a display 406, a network
interface 408, a storage area 130, and a user input interface 412. A processor 404 may be
any commercially available processor and may include dual microprocessors and multi-
processor architectures. The display 406 may be any type of visual display unit. The network interface 408 facilitates wired or wireless communications between the computing device 102 and a communications framework. The user input interface 412 facilitates communications between the computing device 102 and input devices, such as a keyboard, mouse, etc.

[0046] The storage area may be composed of a memory 410 and a storage device 416. The storage device 416 may be magnetic storage, volatile storage, non-volatile storage, optical storage, DVD, CD, floppy disk drive, hard disk drive, and the like. The memory 410 may be any computer-readable storage media that may store executable procedures, applications, and data. The computer-readable media does not pertain to propagated signals, such as modulated data signals transmitted through a carrier wave. It may be any type of memory device (e.g., random access memory, read-only memory, etc.), magnetic storage, volatile storage, non-volatile storage, optical storage, DVD, CD, floppy disk drive, hard disk drive, and the like. The memory 410 may also include one or more external storage devices or remotely located storage devices. The memory 410 may contain instructions and data as follows:

- an anti-malware program 108;
- an operating system 110;
- system settings 112;
- an infection history 114;
- a restore program 116;
- a restore counter 118;
- a malware dictionary 122;
- signatures 124;
- updates 126; and
- various other applications and data 414.

[0047] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

[0048] Various embodiments may be implemented using hardware elements, software elements, or a combination of both. Examples of hardware elements may include devices,
components, processors, microprocessors, circuits, circuit elements, integrated circuits, application specific integrated circuits, programmable logic devices, digital signal processors, field programmable gate arrays, memory units, logic gates and so forth. Examples of software elements may include software components, programs, applications, computer programs, application programs, system programs, machine programs, operating system software, middleware, firmware, software modules, routines, subroutines, functions, methods, procedures, software interfaces, application program interfaces, instruction sets, computing code, code segments, and any combination thereof. Determining whether an embodiment is implemented using hardware elements and/or software elements may vary in accordance with any number of factors, such as desired computational rate, power levels, bandwidth, computing time, load balance, memory resources, data bus speeds and other design or performance constraints, as desired for a given implementation.

Some embodiments may comprise a storage medium to store instructions or logic. Examples of a storage medium may include one or more types of computer-readable storage media capable of storing electronic data, including volatile memory or non-volatile memory, removable or non-removable memory, erasable or non-erasable memory, writeable or re-writeable memory, and so forth. Examples of the logic may include various software elements, such as programs, procedures, module, applications, code segments, program stacks, middleware, firmware, methods, routines, and so on. In an embodiment, for example, a computer-readable storage medium may store executable computer program instructions that, when executed by a processor, cause the processor to perform methods and/or operations in accordance with the described embodiments. The executable computer program instructions may be implemented according to a predefined computer language, manner or syntax, for instructing a computer to perform a certain function. The instructions may be implemented using any suitable high-level, low-level, object-oriented, visual, compiled and/or interpreted programming language.
CLAIMS

1. A computer-implemented method, comprising:
   executing at least one system restore in a computing device;
   analyzing a plurality of conditions to determine if malware detection is needed;
   determining a level of malware detection upon determining that malware detection
   is needed; and
   performing the malware detection at the determined level.

2. The computer-implemented method of claim 1, the determining step further
   comprising:
   setting a deep scan level of malware detection if a number of previously-executed
   restore operations exceed a threshold.

3. The computer-implemented method of claim 1, the determining step further
   comprising:
   setting a deep scan level of malware detection when a critical system setting is not
   enabled.

4. The computer-implemented method of claim 1, the determining step further
   comprising:
   setting a deep scan level of malware detection when a number of previous
   infections exceeds a threshold.

5. The computer-implemented method of claim 1, the determining step further
   comprising:
   contacting an application store for data pertaining to previously obtained
   applications downloaded to the computing device; and
   setting a deep scan level of malware detection when a known malware program
   was previously downloaded to the computing device.

6. A computer-implemented system, comprising:
   a memory including an operating system; and
   at least one processor programmed to:
   restore one or more files of the operating system,
   analyze a plurality of conditions affecting operation of the system,
   determine whether malware detection is warranted based on occurrence of
   the conditions; and
   determines a level of malware detection based on occurrence of the
   conditions.
7. The system of claim 6, wherein the at least one processor is further programmed to:
   utilize a restoration history to determine whether malware detection is warranted.

8. The system of claim 6, wherein the at least one processor is further programmed to:
   utilize critical settings associated with an operating system to determine whether malware detection is warranted.

9. The system of claim 6, wherein the at least one processor is further programmed to:
   utilize an infection history to determine whether malware detection is warranted.

10. The system of claim 6, wherein the at least one processor is further programmed to:
    utilize information provided by an application store to determine whether malware detection is warranted.
RESTORE PROGRAM INITIATED TO PERFORM RESTORATION

Determine level of scanning needed, if any, and perform highest level of scanning

Remediate any infections that may have been found

FIG. 2
CHECK RESTORE COUNTER TO DETERMINE LEVEL OF SCANNING
CHECK SYSTEM SETTINGS TO DETERMINE LEVEL OF SCANNING, IF ANY
CHECK INFECTION HISTORY TO DETERMINE LEVEL OF SCANNING, IF ANY
CHECK APP STORE SERVER TO DETERMINE LEVEL OF SCANNING, IF ANY
RETURN

FIG. 3
FIG. 4

DEEP SCAN LEVEL DETERMINED

RETURN

RESTORE COUNTER > THRESHOLD?

YES

USER FILE LEVEL DETERMINED

NO

FIG. 5

DEEP SCAN LEVEL DETERMINED

RETURN

ARE ALL CRITICAL SETTINGS ENABLED?

YES

ANALYZE SYSTEM SETTINGS

NO

IS THE NUMBER OF NON-CRITICAL SETTINGS DISABLED > THRESHOLD?

YES

NO
FIG. 8
INTERNATIONAL SEARCH REPORT

INTERNATIONAL SEARCH REPORT

PCT/US2012/069978

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/56(2013.01)i, G06F 15/16(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 21/56; H04W 84/10; G06F 9/445; G06F 17/00; G06F 21/00; G08B 23/00

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic database consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: malware, detection, restore, scan, virus, monitor, malicious

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>See pages 10-13.</td>
<td>2-5, 7-10</td>
</tr>
<tr>
<td>A</td>
<td>[It] 10-2009-0050503 A (LENEC0 CORPORATION) 20 May 2009 See paragraphs 12-13, 21, 30; and figure 3.</td>
<td>1-10</td>
</tr>
<tr>
<td>A</td>
<td>US 2009-0037496 AI (BENEDICT T. CHONG et al.) 05 February 2009 See paragraphs 50-52; and figure 3.</td>
<td>1-10</td>
</tr>
<tr>
<td>A</td>
<td>US 2008-0301796 AI (VLADIMIR HILOSTOV et al.) 04 December 2008 See paragraphs 32-34; and figure 1.</td>
<td>1-10</td>
</tr>
<tr>
<td>A</td>
<td>US 2011-0214186 AI (RESHMA P. KHILNANI et al.) 01 September 2011 See paragraphs 34-37; and figures 4, 6.</td>
<td>1-10</td>
</tr>
</tbody>
</table>

Further categories of cited documents:

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

*E* earlier application or patent but published on or after the international filing date

*L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

*O* document referring to an oral disclosure, use, exhibition or other means

*P* document published prior to the international filing date but later than the priority date claimed

See patent family annex.

Date of the actual completion of the international search

18 April 2013 (18.04.2013)

Date of mailing of the international search report

19 April 2013 (19.04.2013)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsagyo-ro, Seo-gu, Daejeon Metropolitan City 305-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

BYUN, Sung Cheal

Telephone No. 82-42-481-8262

Form PCT/ISA/210 (second sheet) (My 2009)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>KR 10-2009-0050503 A1</td>
<td>20.05.2009</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2008-150707 A3</td>
<td>22.01.2009</td>
</tr>
<tr>
<td>US 2011-0214186 A1</td>
<td>01.09.2011</td>
<td>CN 101681407 A</td>
<td>24.03.2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 2156357 A1</td>
<td>24.02.2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 2156357 A4</td>
<td>01.02.2012</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2010-527075 A</td>
<td>05.08.2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2011-0078796 A1</td>
<td>31.03.2011</td>
</tr>
</tbody>
</table>