(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 April 2006 (13.04.2006)

PCT

(10) International Publication Number
WO 2006/039352 A2

(51) International Patent Classification:
G06F 12/14 (2006.01)

(21) International Application Number:
PCT/US2005/034874

(22) International Filing Date:
28 September 2005 (28.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

10/956,573	1 October 2004 (01.10.2004)	US
10/956,578	1 October 2004 (01.10.2004)	US
10/956,574	1 October 2004 (01.10.2004)	US

(71) Applicant *(for all designated States except US)*: **WEBROOT SOFTWARE, INC.** [US/US]; 2560 55th Street, Boulder, CO 80301 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **THOMAS, Steve** [US/US]; 2420 Panorama Ave., Boulder, CO 80304 (US). **GREENE, Michael, P.** [US/US]; 1760 Linden Avenue, Boulder, CO 80304 (US). **STOWERS, Bradley, D.** [US/US]; 2558 Jarett Drive, Mead, CO 80542 (US). **BARTON, Kevin** [US/US]; 10253 Robb Street, Broomfield, CO 80021 (US). **HERMAN, Jeffery** [US/US]; 5646 Rim Rock Court, Boulder, CO 80301 (US).

(74) Agents: **GALLIANI, William, S.** et al.; Cooley Godward, LLP, One Freedom Square, Reston Town Center, Reston, VA 20190-5601 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

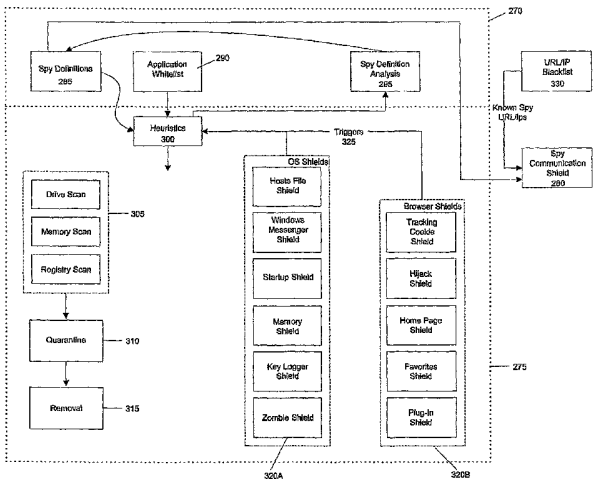
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR PESTWARE DETECTION



(57) Abstract: Methods for monitoring network communications between a protected computer and a remotely-located computer such as a Web server are described. One embodiment is configured to intercept a data packet transmitted from a protected computer. This embodiment then compares the destination address of the data packet against a list of approved destination addresses. When the destination address is included in the list of approved destination addresses, then the packet is delivered to the destination address. If the packet is not addressed to an approved address, then it is evaluated for pestware traces. Embodiments of the invention can also be configured to monitor incoming traffic to a protected computer.

WO 2006/039352 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR PESTWARE DETECTION

COPYRIGHT

FIELD OF THE INVENTION

[0002] The present invention relates to computer system management. In particular, but not by way of limitation, the present invention relates to systems and methods for controlling pestware or malware.

BACKGROUND OF THE INVENTION

[0003] Personal computers and business computers are continually attacked by trojans, spyware, and adware, collectively referred to as "malware" or "pestware." These types of programs generally act to gather information about a person or organization—often without the person or organization's knowledge. Some pestware is highly malicious. Other pestware is non-malicious but may cause issues with privacy or system performance. And yet other pestware is actual beneficial or wanted by the user. Wanted pestware is sometimes not characterized as "pestware" or "spyware." But, unless specified otherwise, "pestware" as used herein refers to any program that collects information about a person or an organization.

[0004] Software is available to detect and remove pestware. But as pestware evolves, the software to detect and remove it must also evolve. Accordingly, current techniques and software are not always currently satisfactory and will most certainly not be satisfactory in

the future. Additionally, because some pestware is actually valuable to a user, pestware-detection software should, in some cases, be able to handle differences between wanted and unwanted pestware.

SUMMARY OF THE INVENTION

[0005] Exemplary embodiments of the present invention that are shown in the drawings are summarized below. These and other embodiments are more fully described in the Detailed Description section. It is to be understood, however, that there is no intention to limit the invention to the forms described in this Summary of the Invention or in the Detailed Description. One skilled in the art can recognize that there are numerous modifications, equivalents and alternative constructions that fall within the spirit and scope of the invention as expressed in the claims.

[0006] Embodiments of the present invention include systems and methods for monitoring computer systems for pestware. One embodiment is configured to intercept a data packet transmitted from a protected computer. This embodiment then compares the destination address of the data packet against a list of approved destination addresses. When the destination address is included in the list of approved destination addresses, then the packet is delivered to the destination address. If the packet is not addressed to an approved address, then it is evaluated for pestware traces.

[0007] Another embodiment of the present invention includes a heuristic engine configured to identify repeat pestware activity and configured to block the repeat pestware activity; an operating system pestware shield in communication with the heuristic engine, the operating system pestware shield configured to detect pestware activity and report the pestware activity

to the heuristic engine; and a browser pestware shield in communication with the heuristic engine, the browser pestware shield configured to detect pestware activity and report the pestware activity to the heuristic engine.

[0008] Yet another embodiment includes a pestware shield configured to detect pestware activity on a protected computer; a heuristics engine configured to identify repeat pestware activity; a drive scan module configured to scan files stored on the storage device and to identify pestware in the scanned files; a program memory scan module configured to scan programs running in the program memory of the protected computer and to identify pestware in the scanned programs; a registry scan module configured to identify any attempts to change data in the registry file; and a quarantine module configured to quarantine the pestware identified by either the drive scan module or the program memory module.

[0009] Embodiments of the invention can also be configured to monitor incoming traffic to a protected computer. These and other embodiments are described in more detail herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Various objects and advantages and a more complete understanding of the present invention are apparent and more readily appreciated by reference to the following Detailed Description and to the appended claims when taken in conjunction with the accompanying Drawings wherein:

FIGURE 1 illustrates a block diagram of one implementation of the present invention;

FIGURE 2 is a flowchart of one method for managing pestware;

FIGURE 3 is a flowchart of one method for handling files after they have been identified as pestware;

FIGURE 4 illustrates another method for handling files after they are identified as pestware;

FIGURE 5 illustrates one method of removing pestware from memory through code injection;

FIGURE 6 illustrates another method for managing pestware that is resistant to permanent removal or that cannot be identified for removal; and

FIGURE 7 is a block diagram of another embodiment of the present invention.

DETAILED DESCRIPTION

[0011] Referring now to the drawings, where like or similar elements are designated with identical reference numerals throughout the several views, and referring in particular to FIGURE 1, it illustrates a block diagram 100 of one implementation of the present invention. This implementation includes four components: a detection module 105, a removal module 110, a reporting module 115, and a shield module 120. Each of these modules can be implemented in software or hardware. And if implemented in software, the modules can be designed to operate on any type of computer system including WINDOWS and Linux-based systems Additionally, the software can be configured to operate on personal computers and/or servers. For convenience, embodiments of the present invention are generally described herein with relation to WINDOWS-based systems. Those of skill in the art can easily adapt these implementations for other types of operating systems or computer systems.

[0012] Referring first to the detection module 105, it is responsible for detecting pestware or pestware activity on a protected computer or system. (The term "protected computer" is used to refer to any type of computer system, including personal computers, handheld computers, servers, firewalls, etc.) Typically, the detection module 105 uses pestware definitions to scan

the files that are stored on a computer system or that are running on a computer system. The detection module 105 can also check WINDOWS registry files and similar locations for suspicious entries or activities. Further, the detection module 105 can check the hard drive for third-party cookies.

[0013] Note that the terms "registry" and "registry file" relate to any file for keeping such information as what hardware is attached, what system options have been selected, how computer memory is set up, and what application programs are to be present when the operating system is started. As used herein, these terms are not limited to WINDOWS and can be used on any operating system.

[0014] Pestware and pestware activity can also be identified by the shield module 120, which generally runs in the background on the computer system. Shields, which will be discussed in more detail below, can generally be divided into two categories: those that use definitions to identify known pestware and those that look for behavior common to pestware. This combination of shield types acts to prevent known pestware and unknown pestware from running or being installed on a protected computer.

[0015] Once the detection or shield module (105 and 120) detects software that could be pestware, the pestware files can be removed or at least quarantined to prevent further issues. The removal module 110, in one implementation, quarantines a potential pestware file and offers to remove it. In other embodiments, the removal module 110 can instruct the protected computer to remove the pestware upon rebooting. And in yet other embodiments, the removal module 110 can inject code into pestware that prevents the pestware from restarting or being restarted.

[0016] In many cases, the detection and shield modules (105 and 120) detect pestware by matching files on the protected computer with definitions of pestware, which are collected from a variety of sources. For example, host computers (shown in Figure 7), protected computers and other systems can crawl the Web to actively identify pestware. These systems often download programs and search for exploits. The operation of these exploits can then be monitored and used to create pestware definitions.

[0017] Alternatively, users can report pestware to a host computer using the reporting module 115. And in some implementations, users may report potential pestware activity to the host computer. The host computer can then analyze these reports, request more information from the target computer if necessary, and then form the pestware definition. This definition can then be pushed from the host computer through a network to one or all of the protected computers and/or stored centrally. Alternatively, the protected computer can request that the definition be sent from the host computer for local storage.

[0018] Referring now to Figure 2, it is a flowchart of one method for managing pestware. In this method, the protected computer initially retrieves a pestware definition. The definition could include a representation of a pestware file or it could include suspicious activity for which the protected computer should monitor. (Block 125) This definition can be retrieved from local storage to the protected computer or from storage remote to the protected computer. The protected computer can then identify and retrieve a target file or an executing program (collectively referred to as a "file" for this figure) that should be checked for pestware. (Block 130) This target file can then be compared against the pestware definition. Target files can be any file on a computer system or only particular files based on file type, such as executable files.

[0019] In one implementation, the target file is scanned to determine if it includes an exact or substantial copy of the pestware. Because scanning all files in a computer system in their entirety would take a significant amount of time, one implementation of the comparison function enables staged comparisons. In the first stage, the protected computer scans its files and running programs for a small portion of a known pestware file. And in some embodiments, the protected computer could use hash functions to speed up any comparison. For example, the definition of a particular known pestware file could include a cyclic redundancy code (CRC) of the first 500 bytes of the file. The protected computer could then calculate a CRC for target files on the protected computer. The target file is any file that is being scanned for pestware. Note that the number of bytes used to calculate the CRCs is not relevant as long as the number of bytes is large enough to provide some degree of accuracy when comparing the target file to a pestware definition.

[0020] After both CRCs have been determined, they can be compared. (Block 135) And if they match, then the second stage of verification determines whether the target file is actually pestware. This second stage involves a complete or substantially complete comparison between the pestware definition and the target file. Generating a hash for both the definition and the file provides one method for comparing them. One example of a hashing algorithm that could be used to compare the definition and the target file is MD5. MD5 produces a digital signature for a file that is as unique as a fingerprint is to a person. Thus, if the result of the MD5 hash is the same for both the target file and the pestware definition, then the two files are virtually guaranteed to be equivalent—meaning that the target file is very likely pestware. (Block 140) And once a file has been identified as pestware, it can be removed. (Block 145)

[0021] Referring now to Figure 3, it illustrates one method of handling files after they have been identified as pestware. In this method, the file is first quarantined. Quarantining usually involves acting on the file to prevent it from executing. In one embodiment, the identified file is compressed, encrypted, and relocated. In other embodiments, it is renamed and relocated. (Blocks 150, 155, and 160) In either embodiment, however, the original location of the file and/or its original state is stored.

[0022] Notably, not all pestware is unwanted or undesirable, and automatic removal is not always an acceptable option for users of these programs. For example, popular file-sharing programs like KAZAA act as wanted spyware. Similarly, the popular GOOGLE toolbar acts as wanted spyware in certain instances. Because users typically want to retain these types of programs, embodiments of the present invention enable the user to selectively identify and retain pestware files. (Block 165) And in certain embodiments, the protected computer can retain a list of approved pestware so that in future sweeps, the computer does not quarantine any pestware included in the list.

[0023] Finally, if the user elects to remove the pestware, it is deleted from the system. (Block 170) But if the user selects to retain the pestware, it is returned to its original form and location. The file can also be flagged as an approved program and added to an approved list. (Block 175)

[0024] Referring now to Figure 4, it shows a method of terminating pestware while the pestware is running. In this implementation, an executing program is identified as pestware that should be terminated. (Blocks 180 and 185) But in certain instances, it is not safe or not possible to terminate the pestware while it is running. The pestware, instead, should be removed upon reboot—before it ever has a chance to execute.

[0025] When these types of pestware programs are identified, instructions can be inserted into the WINDOWS registry file, for example, the "run once" folder, to remove the associated files upon reboot—before the pestware program is started. The removal instruction should be inserted high on the "run once" list in case the pestware is monitoring for removal commands or has its own commands inserted into the registry file. Additionally, the removal instruction should be monitored and protected to prevent the pestware from removing it completely from the registry file.

[0026] Assuming that the pestware program can be safely shut down while it is running, the user is given the option of terminating the program. And if the user elects to terminate the program, the pestware is requested to shut itself down, through, for example a WM_CLOSE message. (Block 190) This request is typically issued through a WINDOWS call. If the pestware program does not terminate itself, then the WINDOWS application program interface (API) is requested to terminate the program. (Block 195) Broadly, if the program will not shut itself down, then the operating system is requested to shut the program down.

[0027] Typically, the operating system can terminate most pestware. But a problem arises when the pestware is associated with a sympathetic program that can restart it. For example, a watcher program can monitor a pestware program, and when the watcher program detects that the pestware program has been terminated, the watcher program could restart it, possibly under a new name. Similarly, when the watcher program is terminated, the pestware program could restart it. These types of mutually-sympathetic programs are difficult for traditional pestware-removal programs to handle. But one implementation of the present invention can address these types of programs by injecting code directly into the pestware programs. (Block 200)

[0028] Referring now to Figure 5, it shows one method of injecting code into pestware. In this implementation, the protected computer attempts to identify the resistant or restarting pestware. And if the pestware itself cannot be identified, then the protected computer should recognize some pestware is restarting itself or is otherwise resistant to removal. (Block 205) Generally, resistant pestware can be identified by storing information about each pestware program removed or quarantined. For example, the protected computer could store the pestware program name, a digital signature for the program, information about the pestware program's activities, or information from the shield module 120. This information, or at least parts of it, can be used to determine if the same pestware program is being removed continuously. And if the protected computer is not being infected continuously from an outside source, then an internal program is likely restarting the pestware program once it is removed. This internal program is called a watcher program.

[0029] If the watcher program can be identified, then it is quarantined along with the pestware program. But if the watcher program cannot be identified, then code injection can be used to block its activities.

[0030] During the code injection process, a termination code program is inserted into each running program or each stored pestware program. (Block 210) This code is often injected into the initialization portion of a file. Generally, this code is a "dll" file that instructs the program upon execution to compare itself to a list of pestware that is being target for removal. This list could all known pestware or just pestware that is known or suspected to restart itself. Additionally, the list of targeted pestware could be stored in the injected code, on the protected computer, or at a remote location.

[0031] In operation, the injected code could instruct a program to check its file name against a list of file names for known pestware or against a list of file names for resistant pestware. Alternatively, the code could instruct the program to call another program to perform the comparison. And in yet another embodiment, the code could instruct the program to call a digital signature function (CRC, hash, etc.) and then compare the digital signature of the program against the signatures for known pestware.

[0032] After the code is injected, the corresponding programs can be terminated. (Block 215) As these programs attempt to restart, the injected code determines whether they should be allowed to completely restart. (Block 220) For example, if the injected code determines that the restarting program matches known pestware or is otherwise suspicious, the injected code can prevent the program from continuing its startup process. (Block 225) Alternatively, the injected code can terminate the program. But if the injected code determines that the restarting program is not pestware, the process is permitted to start. And in some cases, the injected code removes itself from non-pestware programs.

[0033] At this point, both sympathetic programs—the watcher and the primary pestware program—should be injected with the termination code. Neither program should be able to restart itself or the other program. Accordingly, both programs can be quarantined and deleted in the normal fashion. (Block 230)

[0034] Referring now to Figure 6, it is another method for managing pestware that is resistant to permanent removal or that cannot be identified for removal. In this implementation, pestware activity is identified. (Block 235) The activity could be identified by the presence of a certain file or by activities on the computer such as changing registry entries. If a pestware program can be identified, then it should be removed. If the program cannot be

identified, then the activity can be blocked. (Block 240) In essence, the symptoms of the pestware can be treated without identifying the cause. For example, if an unknown pestware program is attempting to change the protected computer's registry file, then that activity can be blocked. Both the pestware activity and the countermeasures can be recorded for subsequent diagnosis. (Block 245)

[0035] Next, the protected computer detects further pestware activity and determines whether it is new activity or similar to previous activity that was blocked. (Blocks 250, 255, and 260) For example, the protected computer can compare the pestware activity—the symptoms—corresponding to the new pestware activity with the pestware activity previously blocked. If the activities match, then the new pestware activity can be automatically blocked. (Block 265) And if the file associated with the activity can be identified, it can be automatically removed.

[0036] To assist in creating a definition for unknown pestware, the reporting module (shown in Figure 1) can bundle information about the pestware and pass it back to a host, which can use that information to form a definition. (Block 270) For example, the records about the protected computer's registry file and any attempted changes can be passed to the host. If the host needs additional information, it may request that information from the reporting module. The user of the protected computer could determine how much information is reported to the host.

[0037] Referring now to Figure 7, it illustrates another embodiment of the present invention. This figure illustrates the host system 270, the protected computer, and an enterprise-protection system 280. The enterprise-protection system 280 could also be used as an

individual consumer product. And in these instances, the consumer could be operating a firewall or firewall-type application.

[0038] The host system 270 can be integrated onto a server-based system or arranged in some other known fashion. The host system 270 could include pestware definitions 285, which include both definitions and characteristics common to pestware. The host system 270 could also include a list of potentially acceptable pestware. This list is referred to as an application white list 290. Applications such as the GOOGLE toolbar and KAAZA could be included in this list. A copy of this list could also be placed on the protected computer where 275 it could be customized by the user. Additionally, the host system 270 could include a pestware analysis engine 295. This engine is configured to receive snapshots of all or portions of a protected computer 275 and identify the activities being performed by pestware. For example, the analysis engine 295 could receive a copy of the registry files for a protected computer that is running pestware. Typically, the analysis engine 295 receives its information from the heuristics engine 300 located on the protected computer 275.

[0039] The pestware-protection functions operating on the protected computer are represented by the sweep engine 305, the quarantine engine 310, the removal engine 315, the heuristic engine 300, and the shields 320. And in this implementation, the shields 320 are divided into the operating system shields 320A and the browser shields 320B. All of these engines can be implemented in a single software package or in multiple software packages.

[0040] The basic functions of the sweep, quarantine, and removal engines was discussed above. To repeat, however, these three engines compare files and registry entries on the protected computer against known pestware definitions and characteristics. When a match is found, the filed is quarantined and removed.

[0041] The shields 320 are designed to watch for pestware and for typical pestware activity and includes two types of shields: behavior-monitoring shields and definition-based shields. In some implementations, these shields can also be grouped as operating-system shields 320A and browser shields 320B.

[0042] The browser shields 320B monitor a protected computer for certain types of activities that generally correspond to pestware behavior. Once these activities are detected, the shield gives the user the option of terminating the activity or letting it go forward. The definition-based shields actually monitor for the installation or operation of known pestware. These shields compare running programs, starting programs, and programs being installed against definitions for known pestware. And if these shields identify known pestware, the pestware can be blocked or removed. Each of these shields is described below.

[0043] Favorites Shield -- The favorites shield monitors for any changes to a browser's list of favorite Web sites. If an attempt to change the list is detected, the shield presents the user with the option to approve or terminate the action.

[0044] Browser-Hijack Shield -- The browser-hijack shield monitors the WINDOWS registry file for changes to any default Web pages. For example, the browser-hijack shield could watch for changes to the default search page stored in the registry file. If an attempt to change the default search page is detected, the shield presents the user with the option to approve or terminate the action.

[0045] Host-File Shield – The host-file shield monitors the host file for changes to DNS addresses. For example, some pestware will alter the address in the host file for yahoo.com to point to an ad site. Thus, when a user types in yahoo.com, the user will be redirected to the

ad site instead of yahoo's home page. If an attempt to change the host file is detected, the shield presents the user with the option to approve or terminate the action.

[0046] Cookie Shield – The cookie shield monitors for third-party cookies being placed on the protected computer. These third-party cookies are generally the type of cookie that relay information about Web-surfing habits to an ad site. The cookie shield can automatically block third-party cookies or it can presents the user with the option to approve the cookie placement.

[0047] Homepage Shield – The homepage shield monitors the identification of a user's homepage. If an attempt to change that homepage is detected, the shield presents the user with the option to approve or terminate the action.

[0048] Common-ad-site Shield – This shield monitors for links to common ad sites, such as doubleclick.com, that are embedded in other Web pages. The shield compares these embedded links against a list of known ad sites. And if a match is found, then the shield replaces the link with a link to the local host or some other link. For example, this shield could modify the hosts files so that IP traffic that would normally go to the ad sites is redirected to the local machine. Generally, this replacement causes a broken link and the ad will not appear. But the main Web page, which was requested by the user, will appear normally.

[0049] Plug-in Shield – This shield monitors for the installation of plug-ins. For example, the plug-in shield looks for processes that attach to browsers and then communicate through the browser. Plug-in shields can monitor for the installation of any plug-in or can compare a

plug-in to a pestware definition. For example, this shield could monitor for the installation of INTERNET EXPLORER Browser Help Objects

[0050] Referring now to the operating system shields 320A, they include the zombie shield, the startup shield, and the WINDOWS-messenger shield. Each of these is described below.

[0051] Zombie shield -- The zombie shield monitors for pestware activity that indicates a protected computer is being used unknowingly to send out spam or email attacks. The zombie shield generally monitors for the sending of a threshold number of emails in a set period of time. For example, if ten emails are sent out in a minute, then the user could be notified and user approval required for further emails to go out. Similarly, if the user's address book is accesses a threshold number of times in a set period, then the user could be notified and any outgoing email blocked until the user gives approval. And in another implementation, the zombie shield can monitor for data communications when the system should otherwise be idle.

[0052] Startup shield – The startup shield monitors the run folder in the WINDOWS registry for the addition of any program. It can also monitor similar folders, including Run Once, Run OnceEX, and Run Services in WINDOWS-based systems. And those of skill in the art can recognize that this shield can monitor similar folders in Unix, Linux, and other types of systems. Regardless of the operating system, if an attempt to add a program to any of these folders or a similar folder, the shield presents the user with the option to approve or terminate the action.

[0053] WINDOWS-messenger shield -- The WINDOWS-messenger shield watches for any attempts to turn on WINDOWS messenger. If an attempt to turn it on is detected, the shield presents the user with the option to approve or terminate the action.

[0054] Moving now to the definition-based shields, they include the installation shield, the memory shield, the communication shield, and the key-logger shield. And as previously mentioned, these shields compare programs against definitions of known pestware to determine whether the program should be blocked.

[0055] Installation shield – The installation shield intercepts the CreateProcess operating system call that is used to start up any new process. This shield compares the process that is attempting to run against the definitions for known pestware. And if a match is found, then the user is asked whether the process should be allowed to run. If the user blocks the process, steps can then be initiated to quarantine and remove the files associated with the process.

[0056] Memory shield – The memory shield is similar to the installation shield. The memory-shield scans through running processes matching each against the known definitions and notifies the user if there is a spy running. If a running process matches a definition, the user is notified and is given the option of performing a removal. This shield is particularly useful when pestware is running in memory before any of the shields are started.

[0057] Communication shield – The communication shield 280 scans for and blocks traffic to and from IP addresses associated with a known pestware site. The IP addresses for these sites can be stored on a URL/IP blacklist 330. And in an alternate embodiment, the communication shield can allow traffic to pass that originates from or is addressed to known

good sites as indicated in a whitelist. This shield can also scan packets for embedded IP addresses and determine whether those addresses are included on a blacklist or whitelist.

[0058] The communication shield 280 can be installed directly on the protected computer, or it can be installed at a firewall, firewall appliance, switch, enterprise server, or router. In another implementation, the communication shield checks for certain types of communications being transmitted to an outside IP address. For example, the shield may monitor for information that has been tagged as private.

[0059] The communication shield could also inspect packets that are coming in from an outside source to determine if they contain any pestware traces. For example, this shield could collect packets as they are coming in and will compare them to known definitions before letting them through. The shield would then block any that are tracks associated with known pestware.

[0060] To manage the timely delivery of packages, embodiments of the communication shield can stage different communication checks. For example, the communication shield could initially compare any traffic against known pestware IP addresses or against known good IP addresses. Suspicious traffic could then be sent for further scanning and traffic from or to known pestware sites could be blocked. At the next level, the suspicious traffic could be scanned for communication types such as WINDOWS messenger or IE Explorer. Depending upon a security level set by the user, certain types of traffic could be sent for further scanning, blocked, or allowed to pass. Traffic sent for further processing could then be scanned for content. For example, does the packet related to HTML pages, Javascript, active X objects, etc. Again, depending upon a security level set by the user, certain types of traffic could be sent for further scanning, blocked, or allowed to pass.

[0061] Key-logger shield – The key-logger shield monitors for pestware that captures and reports out key strokes by comparing programs against definitions of known key-logger programs. The key-logger shield, in some implementations, can also monitor for applications that are logging keystrokes—independent of any pestware definitions. In these types of systems, the shield stores a list of known good programs that can legitimately log keystrokes. And if any application not on this list is discovered logging keystrokes, it is targeted for shut down and removal. Similarly, any key-logging application that is discovered through the definition process is targeted for shut down and removal. The key-logger shield could be incorporated into other shields and does not need to be a stand-alone shield.

[0062] Still referring to Figure 7, the heuristics engine 300, it blocks repeat activity and can also notify the host system 270 about reoccurring pestware. Generally, the heuristics engine 270 is tripped by one of the shields (shown as trigger 325). Stated differently, the shields report any suspicious activity to the heuristics engine 300. If the same activity is reported repeatedly, that activity can be automatically blocked or automatically permitted—depending upon the user's preference. The heuristics engine 300 can also present the user with the option to block or allow an activity. For example, the activity could be allowed once, always, or never.

[0063] And in some implementations, any blocked activity can be reported to the host system 270 and in particular to the definition analysis engine 295. The definition analysis engine 295 can use this information to form a new pestware definition or to mark characteristics of certain pestware.

[0064] In conclusion, the present invention provides, among other things, a system and method for managing pestware. Those skilled in the art can readily recognize that numerous

variations and substitutions may be made in the invention, its use and its configuration to achieve substantially the same results as achieved by the embodiments described herein. Accordingly, there is no intention to limit the invention to the disclosed exemplary forms. Many variations, modifications and alternative constructions fall within the scope and spirit of the disclosed invention as expressed in the claims.

WHAT IS CLAIMED IS:

1. A method for monitoring network communications for pestware, the method comprising:

receiving a data packet from a protected computer, the data packet including a destination IP address;

comparing the destination IP address against a list of IP addresses associated with pestware; and

blocking the data packet from being delivered to the destination IP address when the destination IP address matches an entry in the list of IP addresses associated with pestware.

2. The method of claim 1, further comprising:

presenting a user with the option of blocking the data packet from being delivered to the destination IP address when the destination IP address matches an entry in the list of IP addresses associated with pestware; and

responsive to the user selecting to block the data packet, blocking the data packet.

3. The method of claim 1, wherein receiving a data packet from a protected computer comprises:

receiving the data packet at a firewall appliance.

4. A method for blocking pestware activity, the method comprising:

detecting an initial pestware activity on a protected computer;

recording the initial pestware activity;

receiving an instruction from a user of the protected computer to block the initial pestware activity;

blocking the initial pestware activity;

detecting a subsequent pestware activity;

comparing the subsequent pestware activity with the initial pestware activity; and

responsive to the subsequent pestware activity matching the initial pestware activity, automatically blocking the subsequent pestware activity.

5. The method of claim 4, wherein detecting the initial pestware activity comprises:

detecting the initial pestware activity with an operating system shield.

6. The method of claim 4, wherein detecting the initial pestware activity comprises:

detecting the initial pestware activity with a browser shield.

7. The method of claim 4, wherein detecting the subsequent pestware activity comprises:

detecting an alteration of a registry file associated with the protected computer.

8. The method of claim 4, wherein detecting the subsequent pestware activity comprises:

detecting an attempted installation at the protected computer of a plug-in.

9. The method of claim 4, wherein detecting the subsequent pestware activity comprises:

detecting the attempted startup at the protected computer of a pestware program.

10. The method of claim 9, wherein detecting the attempted startup of the pestware program comprises:

comparing an indication of the pestware program with a definition of the pestware program.

11. The method of claim 4, wherein detecting the subsequent pestware activity comprises:

detecting the attempted installation at the protected computer of a pestware program.

12. The method of claim 11, wherein detecting the attempted startup of the pestware program comprises:

comparing an indication of the pestware program with a definition of the pestware program.

13. The method of claim 4, further comprising:

sending data from the protected computer to a host computer, the data indicating the subsequent pestware activity.

14. The method of claim 4, further comprising:

collecting information from the registry file about the subsequent pestware activity; and

sending the collected information from the protected computer to a host computer.

15. The method of claim 4, further comprising:

identifying the process responsible for the subsequent pestware activity; and

sending an identification of the process from the protected computer to a host computer.

16. The method of claim 4, further comprising:

identifying the process responsible for the subsequent pestware activity; and

injecting termination code into the process.

17. The method of claim 4, further comprising:

identifying the file responsible for the subsequent pestware activity; and

injecting termination code into the file.

18. A system for managing pestware, the system comprising:

a pestware shield configured to detect initial pestware activity on a protected computer and a second pestware activity on the protected computer, the protected computer including a storage device, a program memory, and a registry file;

a heuristics engine in communication with the pestware shield, the heuristics engine configured to determine if the second pestware activity is similar to the initial pestware activity;

a drive scan module configured to scan files stored on the storage device and to identify pestware in the scanned files;

a program memory scan module configured to scan programs running in the program memory of the protected computer and to identify pestware in the scanned programs;

a registry scan module configured to identify any attempts to change data in the registry file; and

a quarantine module configured to quarantine the pestware identified by either the drive scan module or the program memory module.

19. The system of claim 18, wherein the pestware shield comprises:

an operating system shield.

20. The system of claim 18, wherein the pestware shield comprises:

a plug-in shield.

21. The system of claim 18, wherein the pestware shield comprises:

a startup shield.

22. The system of claim 18, wherein the pestware shield comprises:

a memory shield.

23. The system of claim 18, wherein the quarantine module is configured to encrypt the identified pestware.

24. The system of claim 18, wherein the quarantine module is configured to relocate the identified pestware from its original storage location on the protected computer to a new storage location.

25. The system of claim 18, wherein the heuristics engine is configured to receive a first trigger indicating the initial pestware activity from the pestware shield and is configure to receive a second trigger from the pestware shield indicating the second pestware activity.

26. The system of claim 18, wherein responsive to determining that the second pestware activity is similar to the initial pestware activity, the heuristics engine is configured to collect information from the registry file of the protected computer and transmit the collected

information to a host system, whereby the host system can generate a definition corresponding to the second pestware activity.

27. The system of claim 18, wherein responsive to determining that the second pestware activity is similar to the initial pestware activity, the heuristics engine is configured to collect information about the state of the protected computer and transmit the collected information to a host system whereby the host system can generate a definition corresponding to the second pestware activity.

28. The system of claim 18, wherein responsive to determining that the second pestware activity is similar to the initial pestware activity, the heuristics engine is configured to collect configuration data corresponding to the protected computer and transmit the collected information to a host system whereby the host system can generate a definition corresponding to the second pestware activity.

29. The system of claim 28, further comprising:

a definitions module configured to receive from the host system and to store the definition corresponding to the second pestware activity.

30. The system of claim 29 wherein the pestware shield is configured to receive the definition corresponding to the second pestware activity and to use the definition to detect subsequent activity.

31. The system of claim 29 wherein the registry scan module is configured to use the definition corresponding to the second pestware activity to detect a file corresponding to the second pestware activity.

32. The system of claim 29 wherein the memory scan module is configured to use the definition corresponding to the second pestware activity to detect a file corresponding to the second pestware activity.

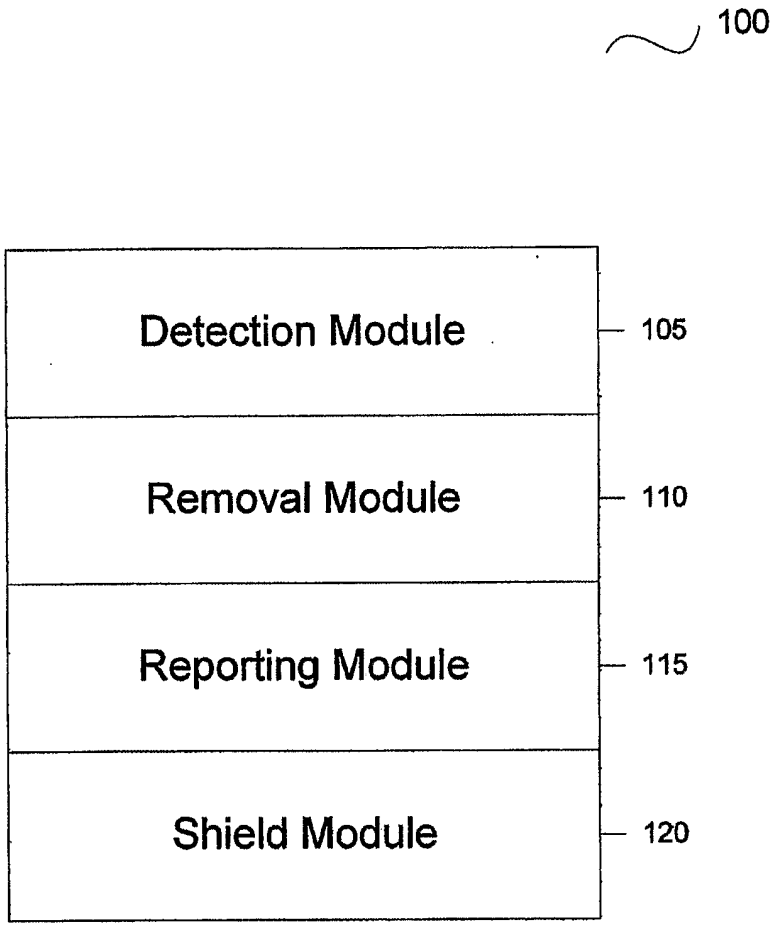

100
Detection Module
105
Removal Module
110
Reporting Module
115
Shield Module
120

Figure 1

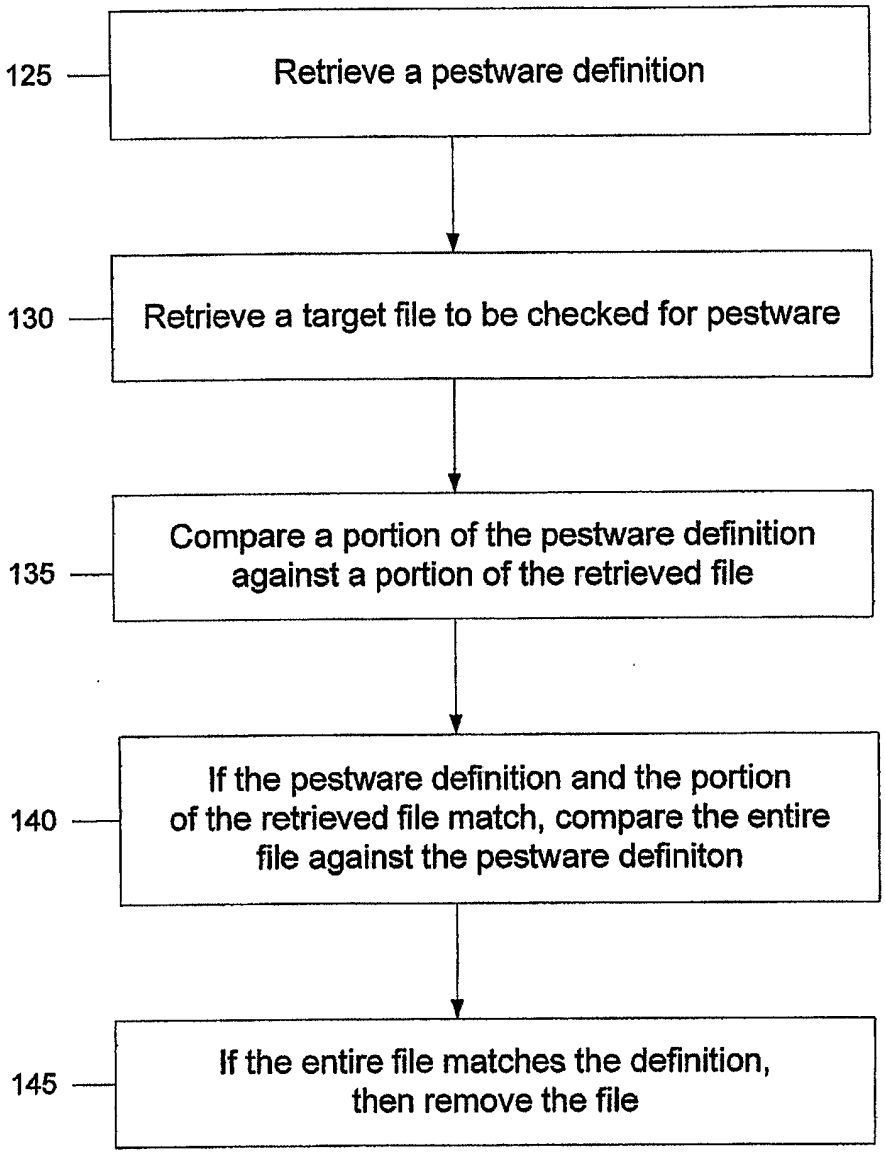

125
Retrieve a pestware definition
130
Retrieve a target file to be checked for pestware
135
Compare a portion of the pestware definition against a portion of the retrieved file
140
If the pestware definition and the portion of the retrieved file match, compare the entire file against the pestware definiton
145
If the entire file matches the definition, then remove the file

Figure 2

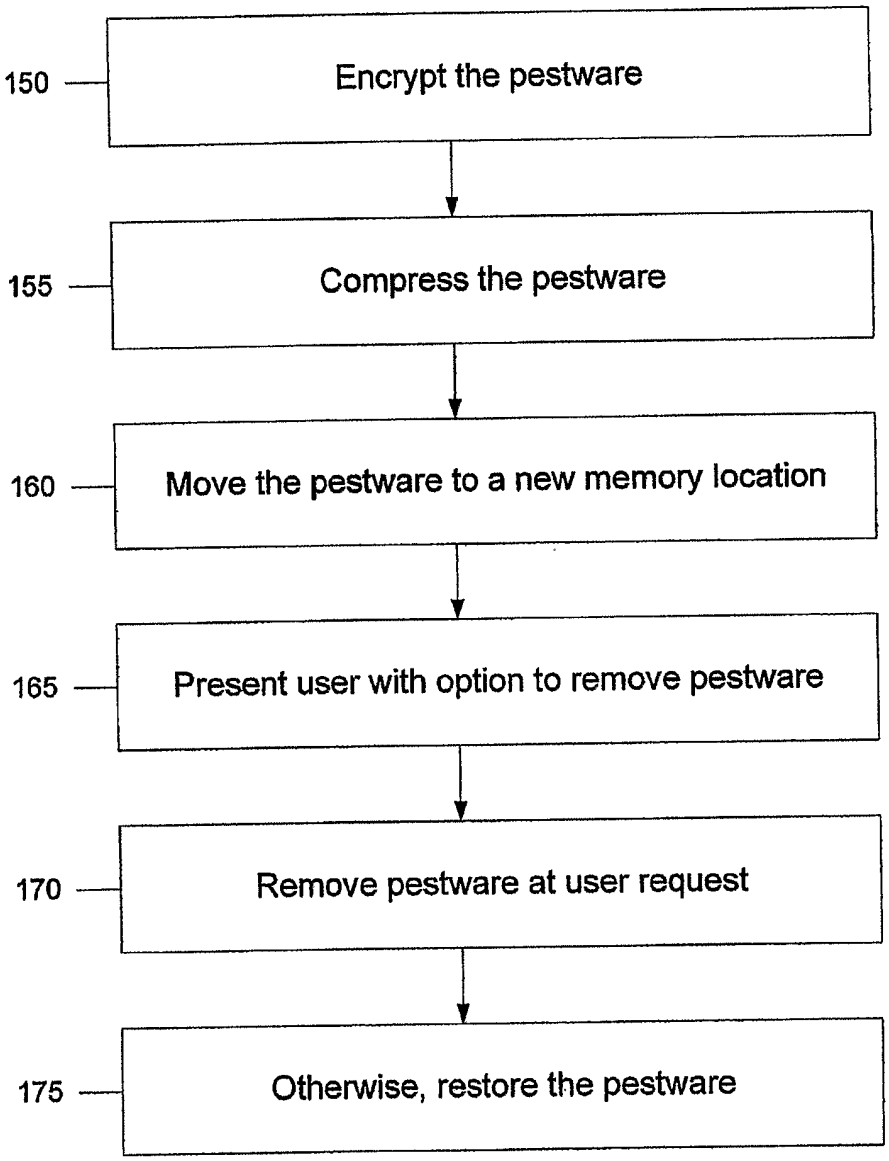

150
Encrypt the pestware
155
Compress the pestware
160
Move the pestware to a new memory location
165
Present user with option to remove pestware
170
Remove pestware at user request
175
Otherwise, restore the pestware

Figure 3

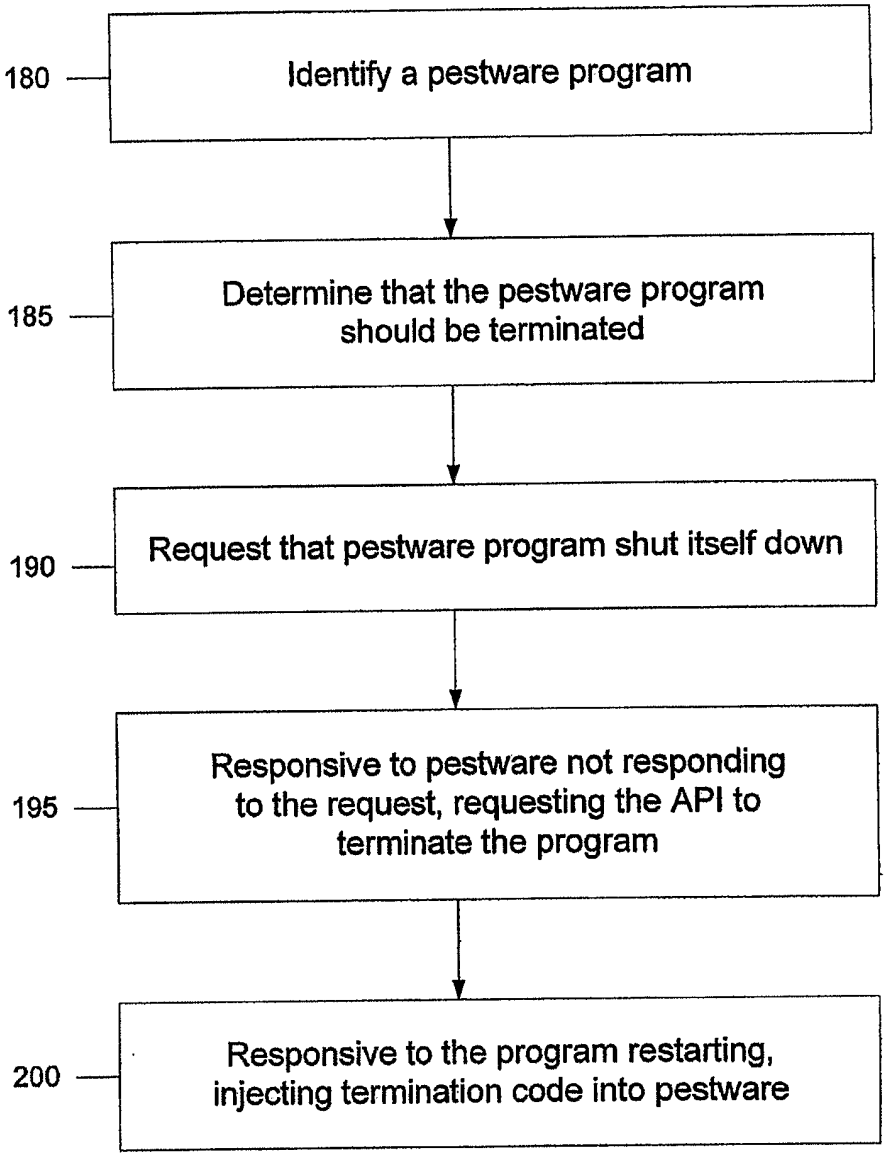

180
Identify a pestware program
185
Determine that the pestware program should be terminated
190
Request that pestware program shut itself down
195
Responsive to pestware not responding to the request, requesting the API to terminate the program
200
Responsive to the program restarting, injecting termination code into pestware

Figure 4

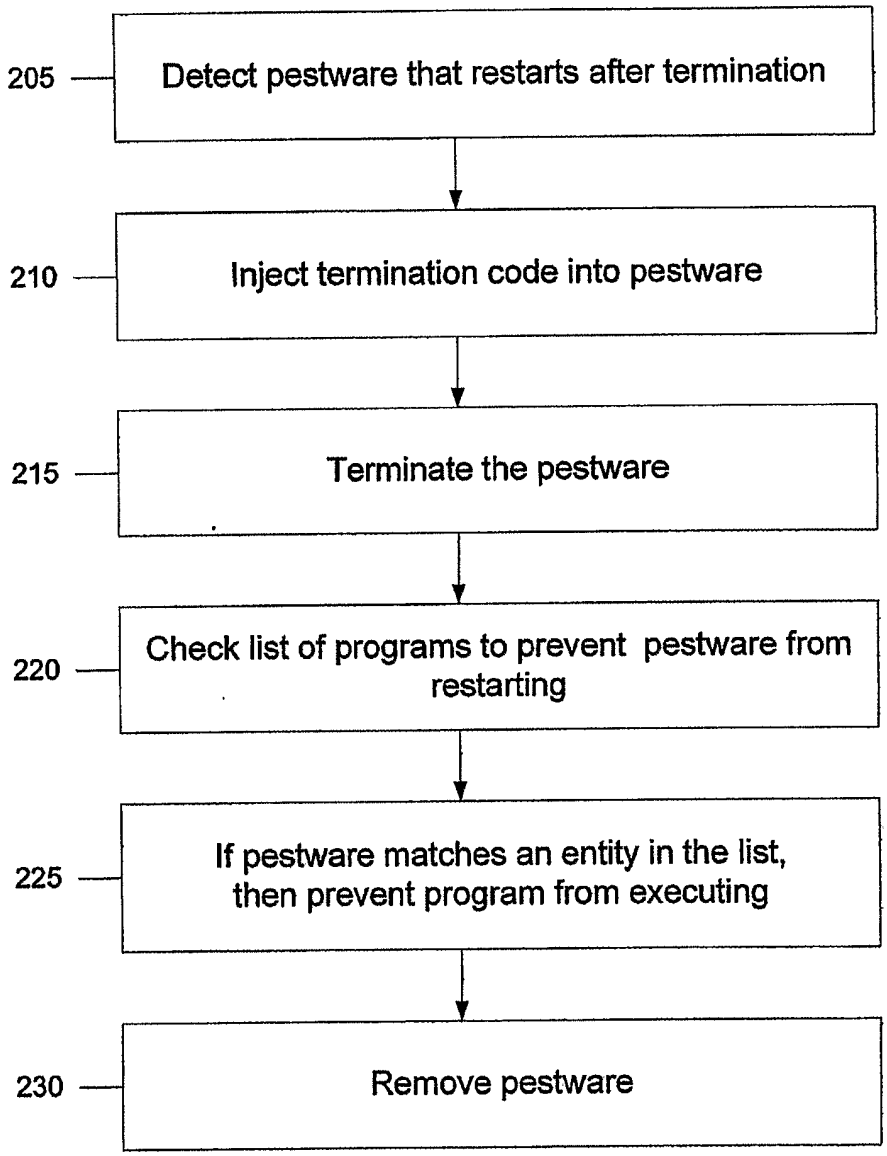

205
Detect pestware that restarts after termination
210
Inject termination code into pestware
215
Terminate the pestware
220
Check list of programs to prevent pestware from restarting
225
If pestware matches an entity in the list, then prevent program from executing
230
Remove pestware

Figure 5

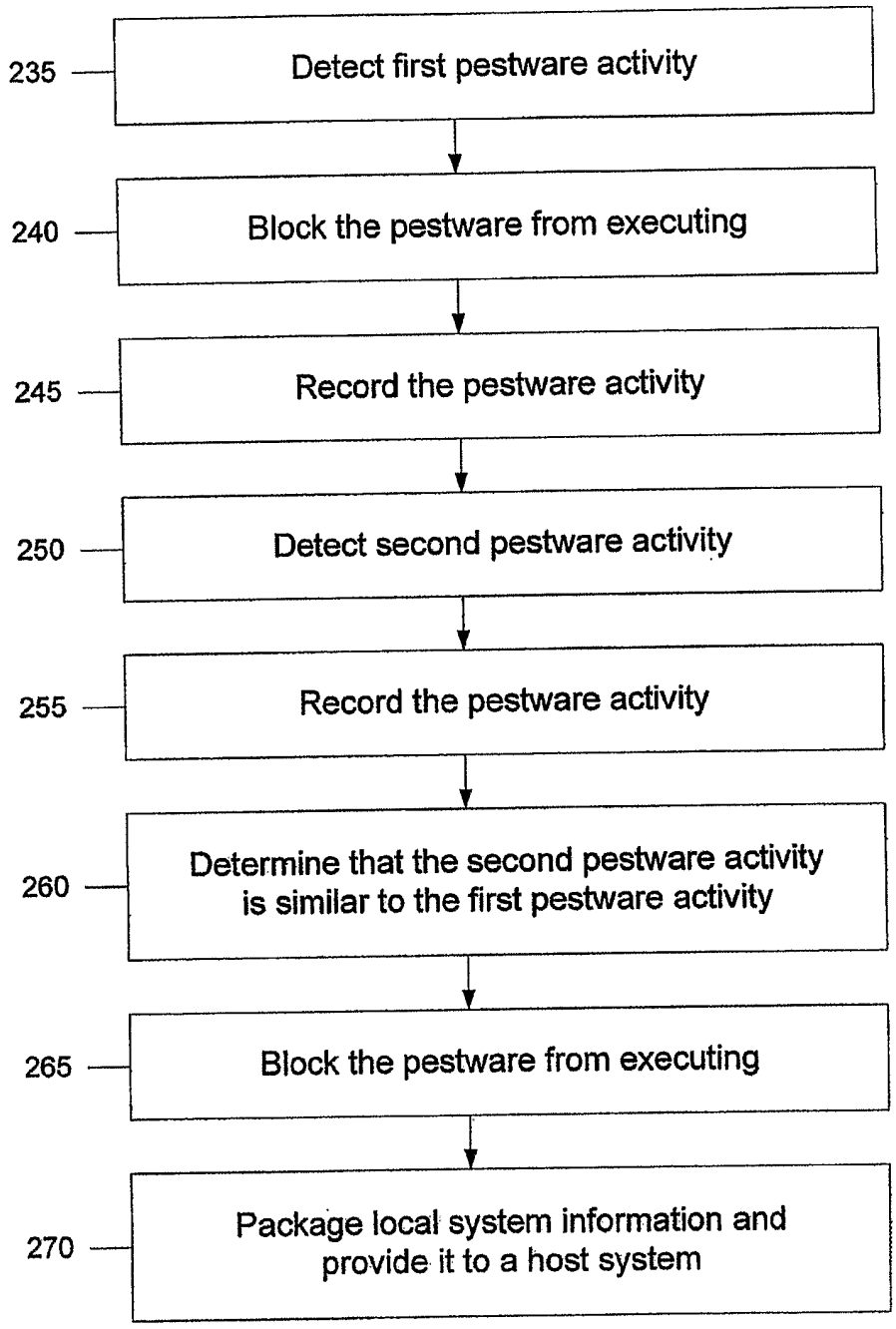

235
Detect first pestware activity
240
Block the pestware from executing
245
Record the pestware activity
250
Detect second pestware activity
255
Record the pestware activity
260
Determine that the second pestware activity is similar to the first pestware activity
265
Block the pestware from executing
270
Package local system information and provide it to a host system

Figure 6

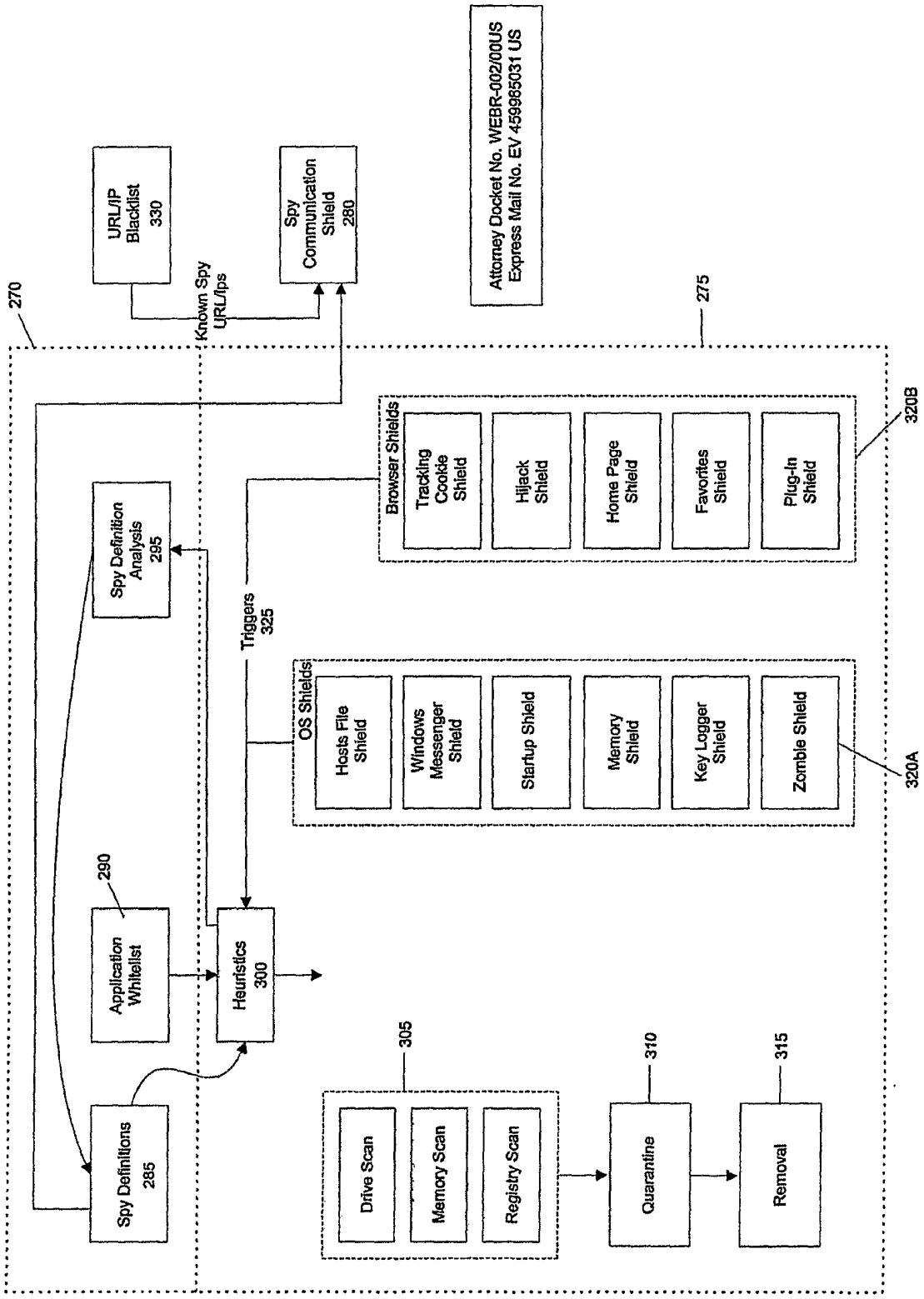

Attorney Docket No. WEBR-002/00US
Express Mail No. EV 459985031 US
URL/IP Blacklist 330
Spy Communication Shield 280
Known Spy URL/Ips
270
275
Spy Definition Analysis 295
Browser Shields
Tracking Cookie Shield
Hijack Shield
Home Page Shield
Favorites Shield
Plug-In Shield
320B
Triggers 325
OS Shields
Hosts File Shield
Windows Messenger Shield
Startup Shield
Memory Shield
Key Logger Shield
Zombie Shield
320A
290
Application Whitelist
Heuristics 300
305
Drive Scan
Memory Scan
Registry Scan
310
Quarantine
315
Removal
Spy Definitions 285

FIGURE 7