US007549055B2

(12) **United States Patent**
Zimmer et al.

(10) **Patent No.: US 7,549,055 B2**
(45) **Date of Patent: Jun. 16, 2009**

(54) **PRE-BOOT FIRMWARE BASED VIRUS SCANNER**

(75) Inventors: **Vincent J. Zimmer**, Federal Way, WA (US); **Michael A. Rothman**, Gig Harbor, WA (US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 880 days.

(21) Appl. No.: **10/441,506**

(22) Filed: **May 19, 2003**

(65) **Prior Publication Data**

US 2004/0236960 A1 Nov. 25, 2004

(51) **Int. Cl.**
G06F 11/30 (2006.01)
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **713/188**; 726/24

(58) **Field of Classification Search** 713/188, 713/2; 726/23, 24
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,475,839 A * 12/1995 Watson et al. 713/2
5,511,184 A * 4/1996 Lin 710/261
5,680,547 A * 10/1997 Chang 709/222
5,802,277 A * 9/1998 Cowlard 726/22
6,338,141 B1 * 1/2002 Wells 726/24
6,347,375 B1 * 2/2002 Reinert et al. 726/24
6,625,730 B1 * 9/2003 Angelo et al. 713/2
6,851,057 B1 * 2/2005 Nachenberg 726/24
6,907,524 B1 * 6/2005 Huntington et al. 713/164
2003/0131248 A1 * 7/2003 Huang 713/188
2004/0205362 A1 * 10/2004 Catherman et al. 713/300

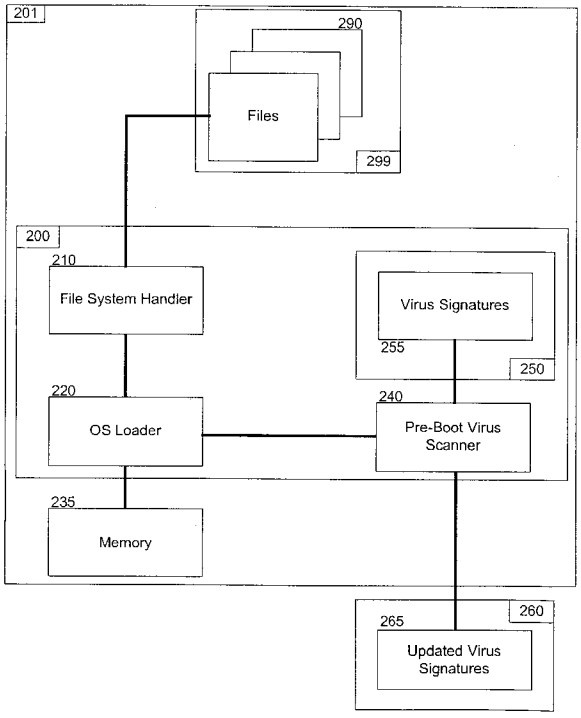
* cited by examiner

Primary Examiner—Matthew B Smithers
Assistant Examiner—Ali S Abyaneh
(74) *Attorney, Agent, or Firm*—Schwabe, Williamson & Wyatt, P.C.

(57) **ABSTRACT**

The present disclosure relates to allowing the utilization of a virus scanner and cleaner that operates primarily in the pre-boot phase of computer operation and, more particularly, to allowing the utilization of a virus scanner and cleaner that operates primarily during the loading of an operating system.

29 Claims, 2 Drawing Sheets



CROWDSTRIKE 1009

Fig. 1

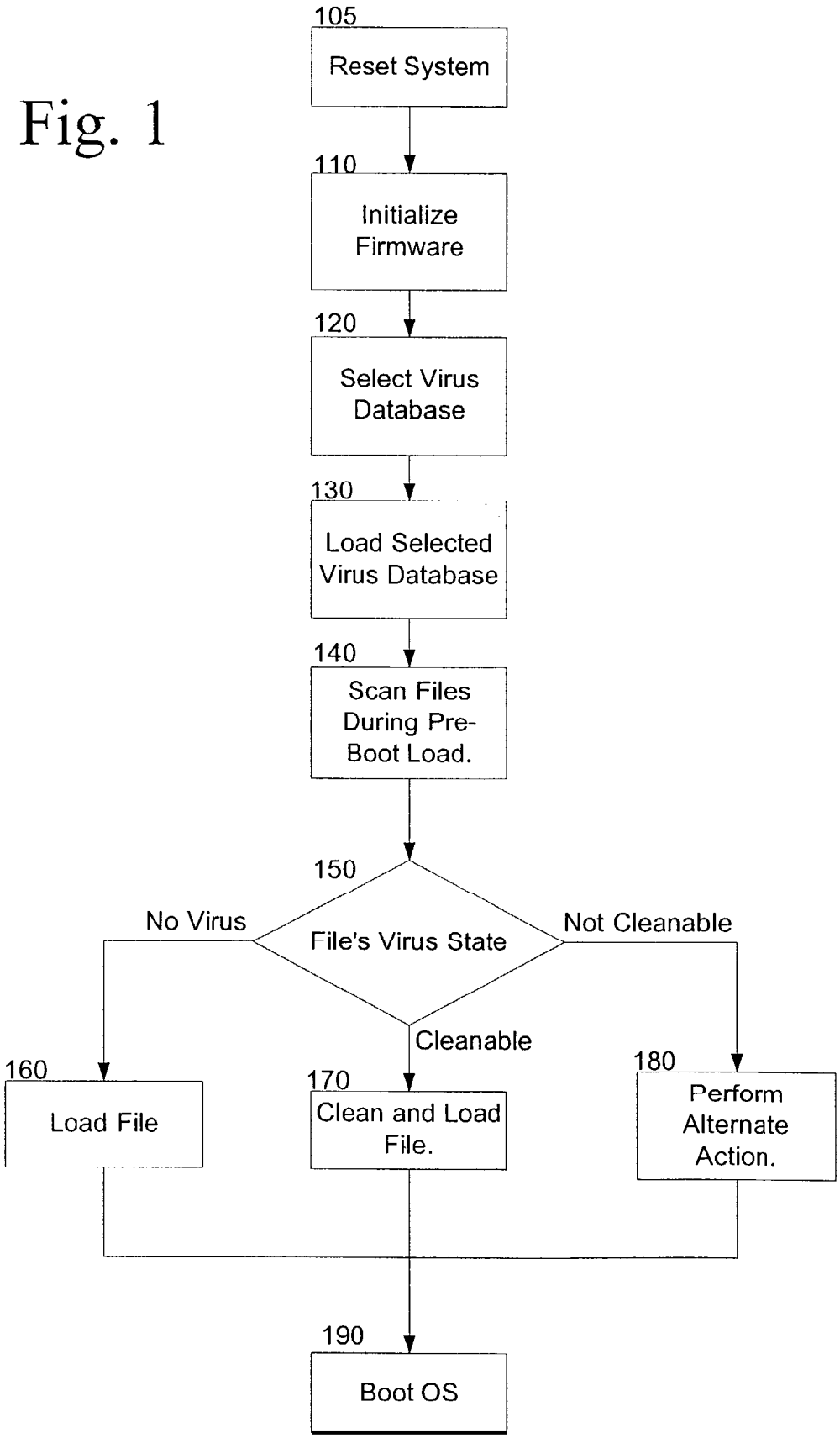
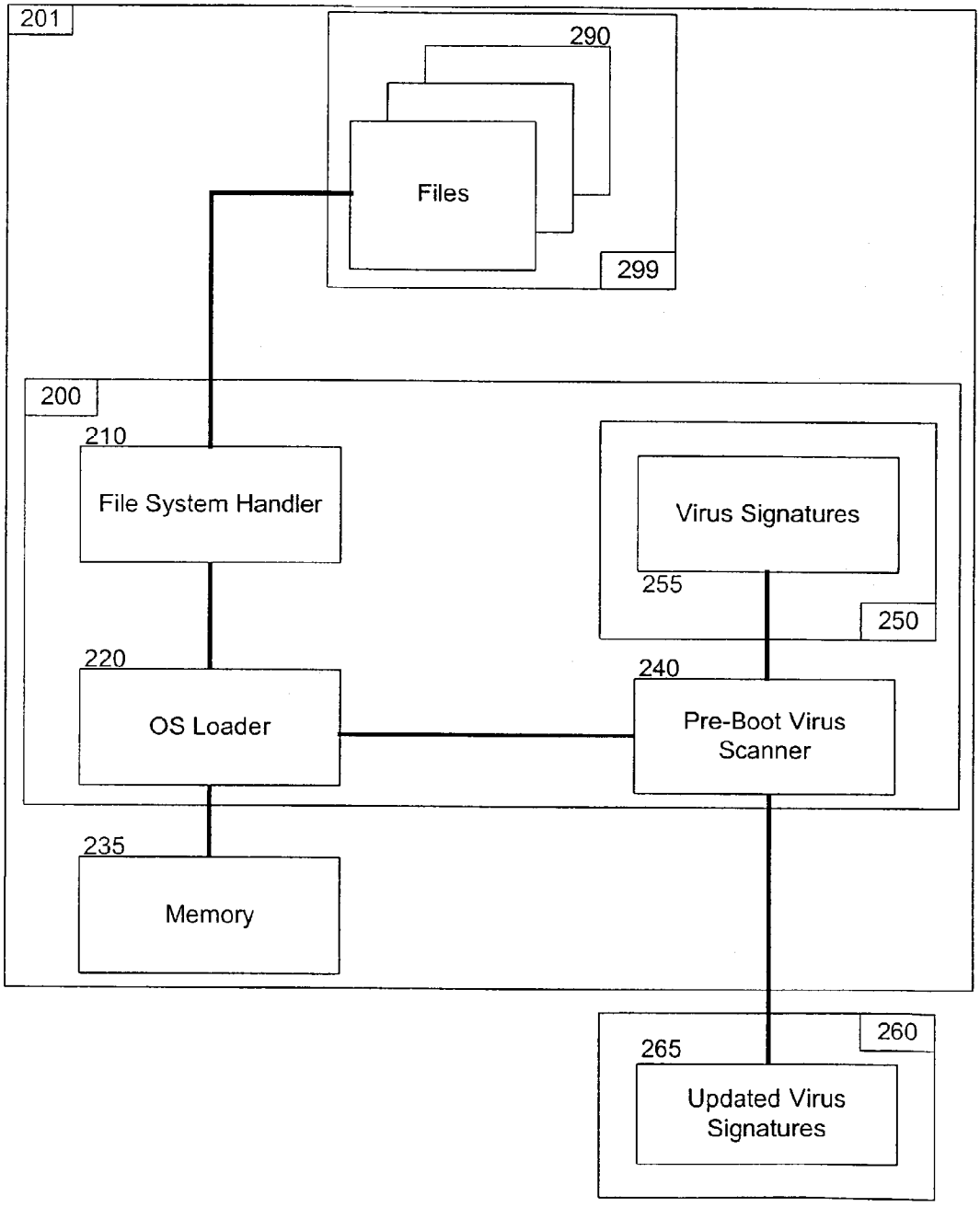

105
Reset System
110
Initialize Firmware
120
Select Virus Database
130
Load Selected Virus Database
140
Scan Files During Pre-Boot Load.
150
File's Virus State
No Virus
Not Cleanable
Cleanable
160
Load File
170
Clean and Load File.
180
Perform Alternate Action.
190
Boot OS

Fig. 2


201
290
Files
299
200
210
File System Handler
Virus Signatures
255
250
220
OS Loader
240
Pre-Boot Virus Scanner
235
Memory
260
265
Updated Virus Signatures

PRE-BOOT FIRMWARE BASED VIRUS SCANNER

BACKGROUND

1. Field

The present disclosure relates to allowing the utilization of a virus scanner and cleaner that operates primarily in the pre-boot phase of computer operation and, more particularly, to allowing the utilization of a virus scanner and cleaner that operates primarily during the loading of an operating system.

2. Background Information

Typically, the operation of a computer or processing system (hereafter, "computer") may be divided into two stages, pre-boot and runtime. The pre-boot process or phase often comprises starting or resetting a computer. When first turned on (cold boot) or reset/rebooted (warm boot), a computer executes the software that loads and starts the computer's operating system and prepares it for use. Thus, the computer can be said to pull itself up by its own bootstraps. The runtime process or phase often occurs after the pre-boot phase and includes the execution of an operating system and other user applications. The runtime phase is typically the phase that users interact with the computer. Thus, the computer can be said to being running application programs. It is contemplated that a computer or processing system includes devices, such as, for example, mobile or stationary computers, personal digital assistants, telecommunication devices and/or similar devices that each include a processor, a storage medium readable or accessible by the processor (including volatile and non-volatile memory and/or storage elements).

In this context, the term "virus" may refer to an intrusive program that infects computer files by inserting in those files copies of itself or a "virus" may refer to a mutation of a binary data image (i.e. a file that has been altered by an intrusive program). The copies may be executed when the file is loaded into memory, allowing them to infect still other files, and so on. Viruses often, but not always, have damaging side effects—sometimes intentionally, sometimes not. For example, some viruses may destroy a computer's hard disk or take up memory space that could otherwise be used by programs. In this context, a virus may also include programs, such as, for example, a destructive program that is disguised as a benign program (i.e. a Trojan Horse), a program that covertly performs an operation without the user's consent or knowledge (e.g. spyware), or other unfriendly programs. During the pre-boot phase virus infected files may be particularly troubling. Normally, in the pre-boot phase all of the software executes in the most privileged processor mode and there is no protection of the memory space. In addition, pre-boot software may be loaded from adapter cards or from across a network.

Typically, computer systems scan, or search files, for viruses during the runtime phase of operation. Common commercial virus scanning techniques include scanning a file system for viruses after the operating system is running. However, by only scanning for viruses during the runtime phase, any viruses which may load or execute during the pre-boot phase or that portion of the runtime phase prior to the running of the virus scanner are unchecked. Furthermore, because current techniques for detecting virus infections run after the operating system is running, the techniques depend and/or are tailored to the specific host operating system. A need, therefore, exists for an improved system or technique for implementing a virus scanning device.

BRIEF DESCRIPTION OF THE DRAWINGS

Subject matter is particularly pointed out and distinctly claimed in the concluding portions of the specification. The disclosed subject matter, however, both as to organization and the method of operation, together with objects, features and advantages thereof, may be best understood by a reference to the following detailed description when read with the accompanying drawings in which:

FIG. **1** is a flowchart illustrating an embodiment of a technique for allowing the utilization of virus scanning in accordance with the disclosed subject matter; and

FIG. **2** is a block diagram illustrating an embodiment of an apparatus and a system that allows the utilization of virus scanning in accordance with the disclosed subject matter.

DETAILED DESCRIPTION

In the following detailed description, numerous details are set forth in order to provide a thorough understanding of the present disclosed subject matter. However, it will be understood by those skilled in the art that the disclosed subject matter may be practiced without these specific details. In other instances, well-known methods, procedures, components, and circuits have not been described in detail so as to not obscure the disclosed subject matter.

FIG. **1** is a flowchart illustrating an embodiment of a technique for allowing the utilization of virus scanning in accordance with the disclosed subject matter. In one embodiment, the technique may include block **105** that illustrates that the system may be reset, and block **190** that illustrates the booting of the operating system (OS). In short, it is contemplated that one embodiment of the disclosed subject matter may occur during the pre-boot phase (i.e., between block **105** and block **190**). However, the disclosed subject matter is not limited to the pre-boot phase and performing a portion of the technique during the runtime phase is contemplated.

In one specific embodiment, the instructions illustrated by blocks **105**, **110**, **120**, **130** & **140** may be executed as a portion of trusted code, such as for example, a hardware boot block. In a specific embodiment, the instructions embodying the disclosed blocks may be substantially complaint with a security scheme such as, for example, the one encapsulated by the Trusted Computing Group (TCG) Main Specification. *Trusted Computing Group (TCG) Main Specification* v 1.1b, Trusted Computing Group Inc. 2003 (hereafter, "the TCG specification"). The hardware boot block or trusted computing base (TCB), in this embodiment, may be a priori be trusted in that the code authorizes subsequent modules that may be loaded. This code may be referred to, in the parlance of the TCG Specification, as a Trusted BIOS Base (TBB).

Block **110** illustrates that, in one embodiment, a substantial portion of the pre-boot firmware may initialize and/or test the integrity of the hardware system. In one embodiment, the hardware system may perform a variation of a power-on self-test (POST) that tests various system components, such as, for example, Random Access Memory (RAM), the disk drives, and the keyboard, to see if they are properly connected and operating. In another embodiment, the hardware system may perform actions substantially compliant with those defined in the Extensible Firmware Interface (EFI) specification. *Extensible Firmware Interface (EFI) Specification*, ver 1.02, Dec. 12, 2000, Intel Corp. (hereafter, "the EFI specification"). However, this is merely one specific embodiment of

the disclosed subject matter. It is further contemplated that in some embodiments other than the one illustrated by FIG. **1**, the process illustrated by block **110** may occur at different sections of the technique, or possibly, not at all.

In a specific embodiment substantially conforming to the EFI specification, the instructions embodying the disclosed subject matter may be formatted in a manner substantially compliant with the Microsoft Portable Executable (PE) format. *Microsoft Portable Executable and Common Object File Format Specification*, Version 6.0, February 1999, Microsoft Corp (hereafter, "the PE format"). In this embodiment, the instructions illustrated by FIG. **1** may run in the most privileged execution mode as a peer to other executable pre-boot agents. It is contemplated that other EFI and non-EFI embodiments may exist that do not conform the PE format.

Block **120** illustrates that a system may select a virus database. In this context, a "virus database" may be a collection of data that includes information to facilitate the identification of viruses or files affected by viruses. Such a database may include virus signatures or checksums that correspond to particular viruses. It is contemplated that a virus database may include a number of separate databases or files. It is further contemplated that the database may be stored in a variety of locations, such as, for example, in firmware, in a locally accessible non-volatile memory, a network accessible medium, or other storage location. In one embodiment, the virus database, may be stored within a portion of firmware.

In one embodiment, block **120** may also illustrate that a determination regarding which of multiple available virus databases will be utilized by the technique. It is contemplated that, in one embodiment, a system may include a virus database stored within, for example, firmware and a second virus database stored within, for example, a hard drive; however, other storage locations and configurations are within the scope of the disclosed subject matter. Block **120** illustrates that one of the two available databases, in this embodiment the firmware database and the hard drive database, may be selected and utilized. It is contemplated that the selection process may be based, upon a number of factors, such as, for example, availability, size of the database, reliability of the database, age of the database, or other possible factors. It is also contemplated that a particular database may be selected for some portion of the technique and another database selected for a second portion of the technique. It is contemplated that in some embodiments, the virus database may be accessible and editable during the runtime phase of operation.

In one embodiment, the virus database may be updated based upon a local policy engine settable by a Firmware Setup program. Alternatively, the virus database may be accessible via a network interface and maintained by a third party, such as, for example, an independent software vendor (ISV), independent BIOS vendor (IBV), an original equipment manufacturer (OEM), or an independent hardware vendor (IHV); however, these are merely a few non-limiting examples. In one embodiment, the virus database may be synchronized with a remote database via a pre-boot networking scheme, such as, for example, the trivial file transfer protocol. *The Trivial File Transfer Protocol, Request For Comments:* 1350, Network Working Group, Rev. 2, July 1992. In another embodiment, an operating system agent, or other runtime agent may retrieve and/or alter the virus database.

Block **130** illustrates that the selected database may be loaded or otherwise made available for reading. In one embodiment, the selected database may be simply read from the firmware. In another embodiment, the selected database may be accessed or loaded via a network interface. It is contemplated that the selected virus database may be accessed via a number of techniques and that the two above embodiments are merely illustrative examples. Such access techniques may include network file transfer protocols —possibly trivial or multicast file transfer protocols—, system bus access to a removable or non-removable drive, and local access to a non-volatile memory element; however, other access techniques are possible and contemplated.

Block **140** illustrates that files may be scanned during the pre-boot loading phase. In this context, "scanning a file for a virus" includes reading at least a portion of the file and asserting if the file includes or has been affected by a virus. This assertion may include the utilization of the virus database selected by the step illustrated by block **120**.

It is contemplated that a select number of files may be scanned, or, conversely, that substantially all files may be scanned. In one embodiment, only the files related to the operating system and similar early running programs may be scanned for viruses. In another embodiment, the pre-boot phase firmware may be aware and able to recognize the structure of some or all of the file systems accessible to the system. Some or all of the files on these file systems may be scanned for viruses.

In one embodiment, these files may include EFI driver files that conform to the PE format, as described above. These EFI driver files may be loaded from a variety of sources, such as, for example, an adapter card or via a network interface. These files may also be one of various operating system loaders, possibly conforming to the PE format and of an EFI subsystem type EFI_APPLICATION. In another EFI specific embodiment, 16-bit BIOS runtime and option ROM code may be encapsulated to conform to the EFI specification and may be scanned for a virus infection.

In one embodiment, the behaviour of the file scanning may be controlled by a virus-scan matrix, or preference file. Such a file may list a number of data types, such as, for example, files, storage elements, or file access mechanisms, and correlate those data types with a specific action, such as, for example, scanning a file system's master boot record, scan files only when loaded into volatile memory, not scanning files, and scanning files before the runtime phase is entered; however, it is noted that the above is merely a few non-limiting examples and other data types and actions are contemplated. Furthermore, in other embodiments, a preference file may include "cause and effect" relationships other than described above.

In one specific example, all of the files loaded into memory during the pre-boot phase may be scanned for a virus as, or immediately prior to, the files are loaded into memory. In a second specific example, a list of the files included in the operating system and loaded during the initial startup of the operating system may be scanned for viruses prior to the booting of the operating system. In a third specific embodiment, substantially all of the files accessible in the pre-boot phase may be scanned to viruses. In a fourth specific embodiment, the firmware may perform any of the operations illustrated by the previous three embodiments during the pre-boor phase, and, in addition, during the runtime phase, the firmware may scan any file during file access, via disk or network, by the operating system. These are merely a few non-limiting specific embodiments of the claimed subject matter and other embodiments are contemplated.

In one embodiment, the contents of a signed file may be scanned even if the file's digital signature denotes that the file has not been altered since it was signed. For example, the signature may denote that the file has not been infected by an outside virus; however, the signature may not denote, or be capable of denoting, that the signed file is itself not a virus. In

a specific embodiment, the signed file may be signed in a manner substantially compliant with that defined in the Trusted Computing Platform Association (TCPA) specification and its derivatives. *Trusted Computing Platform Association (TCPA) Main Specification*, ver 1.1b, 2000, Compaq Computer Corp. et. al. (hereafter, "the TCPA specification"). Alternatively, it is contemplated that, in some embodiments, a signed file may be regarded as virus free and no scanning of the contents of the signed file may be performed.

In an EFI specific embodiment, the virus database and instructions for scanning may operate in the runtime phase in System Management Mode, which is capable of lockable memory protection. These instructions may provide for scanning of the BIOS runtime regions.

Block **150** illustrates that the file's virus state may be ascertained during the scanning illustrated by block **140**. A virus state may include, but is not limited to: a file is not infected with a virus; a file is infected with a virus and the infection may be sufficiently removed, repaired, or cleaned; or a file is infected with a virus and the infection may not be sufficiently removed, repaired, or cleaned. Of course, other file virus states are possible and contemplated and that these above are merely specific illustrative examples.

In one embodiment, information detailing how to clean an infected file may be included in the selected virus database. It is contemplated that in other embodiments, the information needed to clean or repair the file may be stored in a separate file or location. It is also contemplated that a number of actions may be performed based, at least in part, upon the file's virus state. The specific embodiment illustrated by FIG. **1** illustrates three actions; however, other actions are contemplated and within the scope of the claimed subject matter. It is further contemplated that the relationship between file virus state and action performed may be detailed in a virus-scan matrix, referred to in the description of block **140** above.

In one embodiment, illustrated by FIG. **1**, the virus state may include, but is not limited to: a file is not infected with a virus; a file is infected with a virus and the infection may be sufficiently removed, repaired, or cleaned; or a file is infected with a virus and the infection may not be sufficiently removed, repaired, or cleaned. Block **160** illustrates that, in one embodiment, if a file's virus state denotes that the file is not infected it may be loaded into memory. Of course, the illustration of this one specific embodiment, illustrated by block **160**, does not preclude other actions coming within he scope of the claimed subject matter.

Block **170** illustrates that, in one embodiment, a file that is infected with a virus may be repaired and loaded into memory. It is contemplated that in some embodiments, the file may be merely repaired or cleaned and not directly loaded into memory. It is further contemplated that the file may be marked for cleaning and the actual cleaning of the file may be delayed until the file is loaded into memory or during a later phase of operation, such as for example, the runtime phase or after a portion of pre-boot initialization has occurred.

Block **180** illustrates that if a file is infected and is not repairable an alternate action may be performed. Such actions may include, but are not limited to, deleting the file, quarantining the file, marking the file as infected, loading the file normally, or performing another action. It is contemplated that in some embodiments, the action performed may be detailed in a virus-scan matrix, as referred to in the discussion of block **140**.

FIG. **2** is a block diagram illustrating an embodiment of an apparatus **200** and a system **201** that allows the utilization of a virus scanner and cleaner that operates primarily in the pre-boot phase of computer operation in accordance with the disclosed subject matter. It is contemplated that an embodiment of such a system may take many forms, such as for example, everything from small handheld electronic devices, such as personal data assistants and cellular phones, to application-specific electronic devices, such as set-top boxes, digital cameras, and other consumer electronics, to medium sized mobile systems such as notebook, sub-notebook, and tablet computers, to desktop systems, workstations, and servers. However, the system is not limited to the forms described above and other forms are contemplated.

In one embodiment, the hardware system **201** may include apparatus **200** and a plurality of hardware devices or components that are capable of being configured with a group of hardware settings. It is contemplated that the hardware system **201** and apparatus **200** may be capable of performing the technique illustrated by FIG. **1** and described above.

In one embodiment, the apparatus **200** may include a file system handler **210** to access to a file system during the pre-boot phase of computer operation. File system handler may facilitate reading from and/or writing to, during the pre-boot phase, the file(s) included within the file system. It is contemplated that the file system handler may also allow access to the file system during the runtime phase. It is further contemplated that the file system handler may allow access to a variety of file systems such as those used by a variety of operating systems or those stored on a variety of volatile or non-volatile memories. Furthermore, it is contemplated that the file system handler may access the file system locally or remotely, such as, for example, via a network.

In one embodiment, the file system handler **210** may access files **290** from a file system **299**. It is contemplated that the file system may be stored within a non-volatile or volatile memory. It is further contemplated that the file system may be storage across multiple storage devices, such as, for example, a network drive that is physically stored on multiple hard drives; however, that is merely one non-limiting embodiment of the claimed subject matter. In the embodiment illustrated by FIG. **2**, the file system is not part of apparatus **200**. It is contemplated that some embodiments of the apparatus may include the file system.

Operating system loader **220** may facilitate the loading of an operating system into memory **235**. The loading of the operating system may be part of the transition from the pre-boot to runtime phases of operation. It is contemplated that the operating system loader may facilitate the loading of non-operating system files into memory. In one embodiment, the operating system loader may utilize file system handler **210** to read files associated with the operating system from a file system **299**.

Pre-boot virus scanner **240** may scan files before being loaded by the operating system loader **220** to determine if the files have been infected by or are a virus. In one embodiment, the pre-boot virus scanner may perform the actions illustrated by blocks **140**, **150**, **170**, & **180** illustrated in FIG. **1** and described above. It is contemplated that the pre-boot virus scanner may be capable of scanning files during the pre-boot phase and may also be capable of scanning files during the runtime phase. In another embodiment, the pre-boot virus scanner may only be capable of scanning files during the pre-boot phase and not the runtime phase. It is contemplated that the pre-boot virus scanner may be, in one embodiment, a portion or an entire integrated circuit specifically designed to facilitate the virus scanning of files.

Pre-boot virus scanner **240** may utilize a Virus Database **250** to facilitate the scanning of files. In one embodiment, the virus database may be a database as described with reference to block **120** of FIG. **1** described above. It is contemplated that

the virus database may include virus signatures **255**. It is also contemplated that multiple virus databases may exist and that one or more may be utilized by the pre-boot virus scanner. In one embodiment, an additional virus database **260** may be utilized. The additional virus database may, in one embodiment, include updated virus signatures **265**. It is contemplated that the pre-boot virus scanner may select between the two (or more) virus databases, or combine the databases in some fashion. It is further contemplated that the additional database(s) may be stored within or without the apparatus **200**, or even within or without system **201**. It is also contemplated that the database may be accessed utilizing a networking protocol, such as, for example, multicast, trivial file transfer protocol.

The techniques described herein are not limited to any particular hardware or software configuration; they may find applicability in any computing or processing environment. The techniques may be implemented in hardware, software, firmware or a combination thereof. The techniques may be implemented in programs executing on programmable machines such as mobile or stationary computers, personal digital assistants, and similar devices that each include a processor, a storage medium readable or accessible by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices. Program code is applied to the data entered using the input device to perform the functions described and to generate output information. The output information may be applied to one or more output devices.

Each program may be implemented in a high level procedural or object oriented programming language to communicate with a processing system. However, programs may be implemented in assembly or machine language, if desired. In any case, the language may be compiled or interpreted.

Each such program may be stored on a storage medium or device, e.g. compact read only memory (CD-ROM), digital versatile disk (DVD), hard disk, non-volatile memory, magnetic disk or similar medium or device, that is readable by a general or special purpose programmable machine for configuring and operating the machine when the storage medium or device is read by the computer to perform the procedures described herein. The system may also be considered to be implemented as a machine-readable or accessible storage medium, configured with a program, where the storage medium so configured causes a machine to operate in a specific manner. Other embodiments are within the scope of the following claims.

While certain features of the disclosed subject matter have been illustrated and described herein, many modifications, substitutions, changes, and equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes that fall within the true spirit of the disclosed subject matter.

What is claimed is:

1. A processor implemented method comprising:

during a pre-boot phase of a system having a processor, an operating system (OS) loader, a file system handler, and a pre-boot virus scanner, all operated by the processor, and a boot phase following the pre-boot phase,

invoking the OS loader to load an OS,

invoking, by the OS loader, the file system handler instead of the OS,

loading by the file system handler, files associated with a file system of the OS,

scanning the files loaded by the file system handler by the pre-boot virus scanner, the scanning including

attempting by the pre-boot virus scanner to determine if the scanned files are infected with a virus before attempting to store the scanned files into a memory, and

assigning by the pre-boot virus scanner a virus state to the scanned files based at least in part upon whether the at least one scanned file is or are infected with a virus, and

on completion of the scanning, conditionally loading the OS by the OS loader to transition the system into the boot phase, based at least in part on the results of the scanning by the pre-boot virus scanner.

2. The method of claim **1**, further comprising selecting by the processor a virus database, the selected virus database containing information to facilitate said attempt to determine if the at least one scanned file is or are infected with a virus.

3. The method of claim **1**, wherein the file system handler is in compliance with the Extensible Firmware Interface (EFI) specification.

4. The method of claim **1**, wherein the scanning is performed utilizing a device consisting of at least one of the following:

a non-volatile memory;

a volatile memory;

a locally stored file system; and

a network interface.

5. The method of claim **2**, wherein selecting by the processor a virus database includes:

selecting by the processor a subset of one or more virus databases from a plurality virus databases, said selected virus database being one of the subset of one or more virus databases; and

loading by the processor only the subset of one or more virus databases from the plurality of virus databases.

6. The method of claim **2**, wherein attempting by the processor to determine if the at least one scanned file is infected with a virus includes:

utilizing a virus signature that is included in the selected virus database.

7. The method of claim **6**, wherein attempting by the processor to determine if the at least one scanned file is infected with a virus includes:

utilizing by the processor a scanned file having a digital signature and contents;

checking by the processor that the digital signature of the scanned file having the digital signature and contents is valid;

attempting by the processor to determine if the contents of the scanned file having the digital signature and contents is infected with a virus; and

marking by the processor the scanned file having the digital signature and contents as being infected with a virus if a criteria is met;

wherein the criteria is chosen from a group consisting of:

the digital signature being invalid,

the contents of the scanned file appears to be infected with a virus, and

the digital signature being invalid and the contents of the scanned file appears to be infected with a virus.

8. The method of claim **7**, wherein the scanned file having a digital signature includes a digital signature in compliance with the Trusted Computing Platform Alliance (TCPA) specification.

9. The method of claim **1**, further including:

marking by the processor the at least one scanned file as being infected with a virus if a criteria is met; and

attempting by the processor to repair the at least one infected file before the at least one infected file is or are loaded into memory.

10. The method of claim **9**, wherein marking by the processor the at least one scanned file as being infected with a virus if a criteria is met includes:

utilizing by the processor a predefined set of criteria;

utilizing by the processor the selected virus database to determine if the at least one infected file is or are repairable; and

attempting by the processor to repair the at least one infected file before the at least one infected file is or are loaded into memory includes:

attempting by the processor to repair the at least one infected file only if the at least one infected file is or are repairable.

11. The method of claim **1**, further including:

resetting the system, by the processor, to enter the pre-boot phase.

12. The method of claim **1**, further comprising attempting by the processor to repair the at least one scanned file before the at least one scanned file is or are loaded into memory if the at least one scanned file is or are determined to be infected with a virus.

13. An apparatus comprising:

an operating system (OS) loader to load an OS and to load files into a memory during or after a boot phase of the apparatus and to, during a pre-boot phase, invoke a file system handler instead of the OS;

the file system handler to load, during the pre-boot phase, files stored within a file system of the Os; and

a pre-boot virus scanner, coupled to the OS loader and the file system handler, to scan files loaded by the file system handler, including attempting to determine if the scanned files are infected with a virus before attempting to store the scanned files into a memory and assigning a virus state to the scanned files based at least in part upon whether the at least one scanned file is or are infected with a virus

wherein, on completion of the scanning, the OS loader is to conditionally load the OS to transition the system into the boot phase, based at least in part on the results of the scanning by the pre-boot virus scanner.

14. The apparatus of claim **13**, wherein the pre-boot virus scanner is further capable of operating during a runtime phase of the apparatus following the boot phase.

15. The apparatus of claim **13**, further comprising a virus database to facilitate the identification of a virus; and wherein the pre-boot scanner is adapted to utilize the virus database to facilitate the identification of a virus infecting a file.

16. The apparatus of claim **15**, wherein the pre-boot virus scanner is capable of:

selecting the virus database from a plurality of virus databases.

17. The apparatus of claim **13**, wherein the file system handler is in compliance with the Extensible Firmware Interface (EFI) specification.

18. The apparatus of claim **13**, wherein the pre-boot virus scanner is capable of:

scanning a file having a digital signature and a contents;

checking the digital signature of the scanned file is valid;

attempting to determine if the contents of the scanned file is infected with a virus; and

marking the scanned file as being infected with a virus if a criteria is met;

wherein the criteria is chosen from a group consisting of:

the digital signature being invalid,

the contents of the scanned file appears to be infected with a virus, and

the digital signature being invalid and the contents of the scanned file appears to be infected with a virus.

19. The apparatus of claim **18**, wherein the scanned file having a digital signature includes a digital signature in compliance with the Trusted Computing Platform Alliance (TCPA) specification.

20. The apparatus of claim **13**, wherein the pre-boot virus scanner is capable of:

marking the scanned file as being infected with a virus if a criteria is met; and

attempting to repair the infected file before the infected file is loaded into memory.

21. An article, comprising:

a storage medium having a plurality of machine accessible instructions, wherein when the instructions are executed by a processor of a system having an operating system (OS) loader, a file system handler, and a pre-boot virus scanner, all operated by the processor, the instructions provide for:

during a pre-boot phase of the system, prior to a boot phase of the system,

invoking the OS loader to load an OS,

invoking, by the OS loader, the file system handler instead of the OS,

loading by the file system handler, files associated with a file system of the operating system,

scanning the files loaded by the file system handler by the pre-boot virus scanner, the scanning including,

attempting by the pre-boot virus scanner to determine if the scanned files are infected with a virus before attempting to store the scanned files into a memory, and

assigning by the pre-boot virus scanner a virus state to the scanned files based at least in part upon whether the at least one scanned file is or are infected with a virus, and

on completion of the scanning, conditionally loading the OS by the OS loader to transition the system into the boot phase, based at least in part on the results of the scanning by the pre-boot virus scanner.

22. The article of claim **21**, wherein the instructions further provide for:

selecting by the processor a subset of one or more virus databases from a plurality virus databases, said selected virus database being one of the subset of one or more virus databases; and

loading by the processor only the selected subset of one or more virus databases from the plurality of virus databases.

23. The article of claim **21**, wherein the file system handler is in compliance with the Extensible Firmware Interface (EFI) specification.

24. The article of claim **21**, wherein the instructions providing for attempting by the processor to determine if the at least one scanned file is or are infected with a virus includes instructions providing for

utilizing by the processor a virus signature that is included in a virus database.

25. The article of claim **24**, wherein the instructions providing for attempting by the processor to determine if the at least one scanned file is or are infected with a virus includes instructions providing for:

utilizing by the processor a scanned file having a digital signature and a contents;

checking by the processor that the digital signature of the scanned file having the digital signature and contents is valid;

attempting by the processor to determine if the contents of the scanned file having the digital signature and contents is infected with a virus; and

marking by the processor the scanned file having the digital signature and contents as being infected with a virus if a criteria is met;

wherein the criteria is chosen from a group consisting of:

the digital signature being invalid,

the contents of the scanned file appears to be infected with a virus, and

the digital signature being invalid and the contents of the scanned file appears to be infected with a virus.

26. The article of claim **25**, wherein the scanned file having a digital signature includes a digital signature in compliance with the Trusted Computing Platform Alliance (TCPA) specification.

27. The article of claim **21**, further including instructions providing for:

marking by the processor the at least one scanned file as being infected with a virus if a criteria is met; and

attempting by the processor to repair the at least one infected file before the at least one infected file is or are loaded into memory.

28. The article of claim **27**, wherein the instructions providing for marking by the processor the at least one scanned file as being infected with a virus if a criteria is met includes instructions providing for:

utilizing by the processor a predefined set of criteria;

utilizing by the processor a virus database to determine if the at least one infected file is or are repairable; and

attempting by the processor to repair the at least one infected file before the at least one infected file is or are loaded into memory includes:

attempting by the processor to repair the at least one infected file only if the at least one infected file is or are repairable.

29. The article of claim **21**, further including instructions providing for resetting the system by the processor to enter the pre-boot phase.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,549,055 B2
APPLICATION NO. : 10/441506
DATED : June 16, 2009
INVENTOR(S) : Zimmer et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 8, line 30, "a plurality virus databases," should read --a plurality of virus databases,--

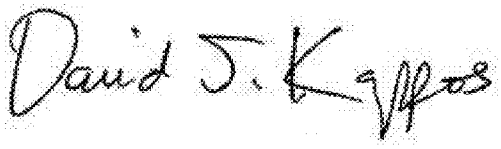
Column 9, line 31, "the Os; and" should read --the OS; and--

Column 9, line 39, "with a virus", should read --with a virus;--

Column 10, line 47, "a plurality virus databases," should read --a plurality of virus databases,--

Column 10, line 59, "providing for", should read --providing for:--

Signed and Sealed this
Twenty-fifth Day of January, 2011



David J. Kappos
Director of the United States Patent and Trademark Office