US008220055B1

(12) **United States Patent**
Kennedy

(10) **Patent No.: US 8,220,055 B1**
(45) **Date of Patent: Jul. 10, 2012**

(54) **BEHAVIOR BLOCKING UTILIZING POSITIVE BEHAVIOR SYSTEM AND METHOD**

(75) Inventor: **Mark K. Kennedy**, Redondo Beach, CA (US)

(73) Assignee: **Symantec Corporation**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 2814 days.

(21) Appl. No.: **10/774,177**

(22) Filed: **Feb. 6, 2004**

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.** **726/25**; 726/22

(58) **Field of Classification Search** 726/22–25
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,076,803	B2 *	7/2006	Bruton et al.	726/23
7,191,252	B2 *	3/2007	Redlich et al.	709/246
2002/0099959	A1 *	7/2002	Redlich et al.	713/201
2003/0061514	A1 *	3/2003	Bardsley et al.	713/201
2004/0083372	A1 *	4/2004	Williamson et al.	713/188
2004/0117648	A1 *	6/2004	Kissel	713/200

FOREIGN PATENT DOCUMENTS

EP	1202228	A1 *	5/2002
GB	2367714	A *	4/2002

* cited by examiner

Primary Examiner — Kim Vu
Assistant Examiner — Randal Moran
(74) *Attorney, Agent, or Firm* — McKay and Hodgson, LLP; Serge J. Hodgson; Sean P. Lewis

(57) **ABSTRACT**

A method includes decreasing a suspicion of a negative action by an application if the application has previously performed a positive action. The positive action is an action that is never or rarely taken by malicious code. In one example, the positive action is use of a user interface element by the application to have a user interaction with a user of the computer system. By taking into consideration the positive action by the application, the occurrence of false positives is minimized.

21 Claims, 2 Drawing Sheets

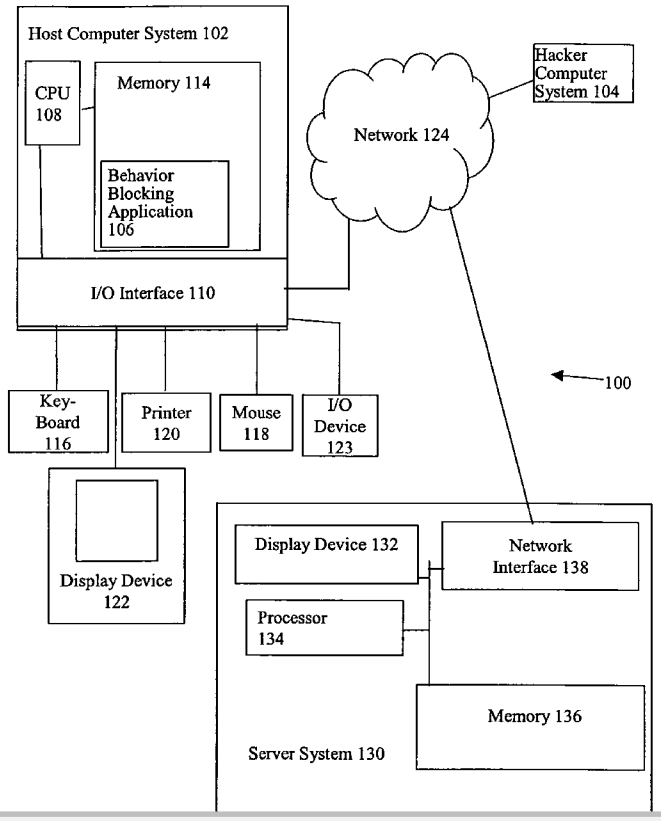


FIG. 1

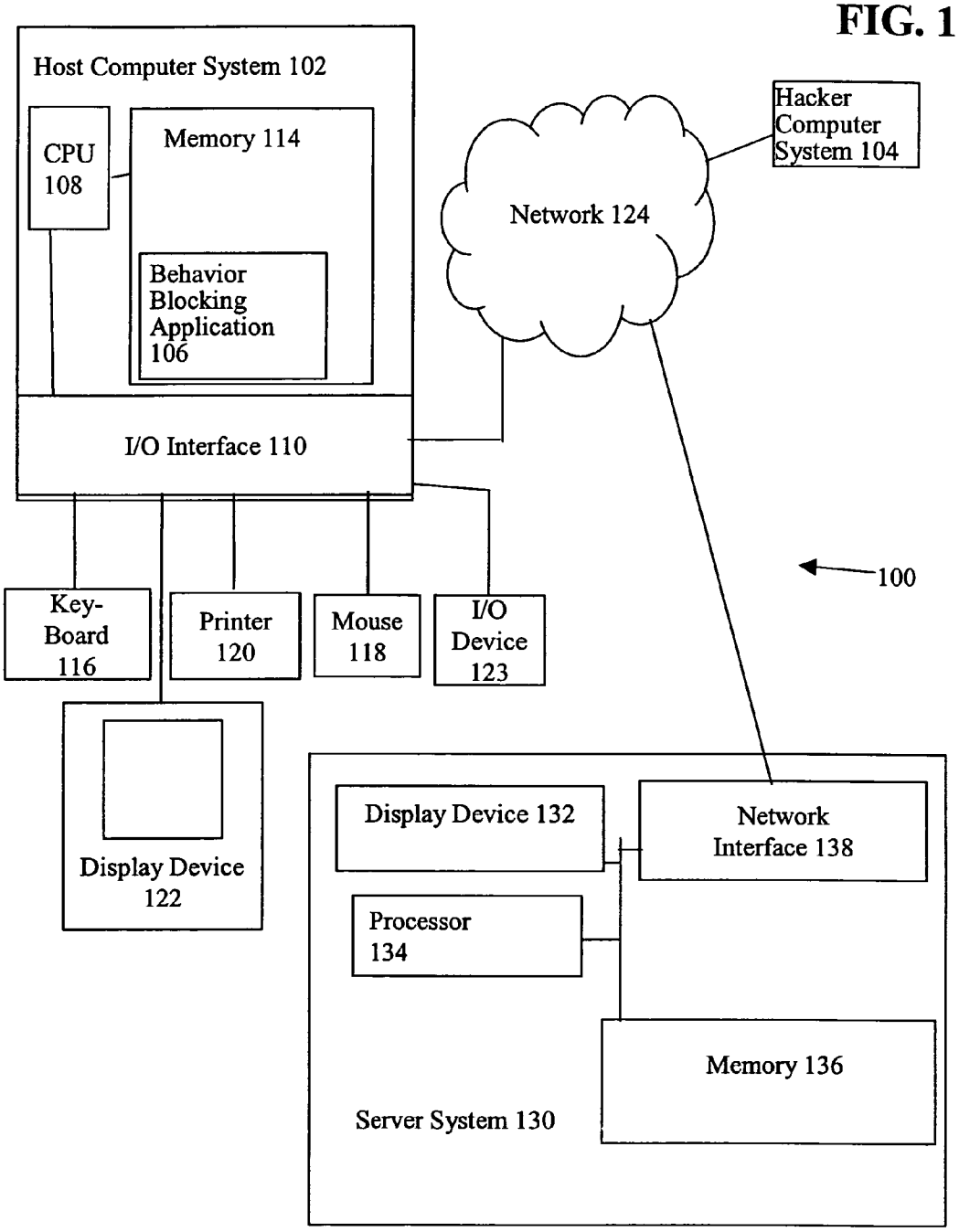
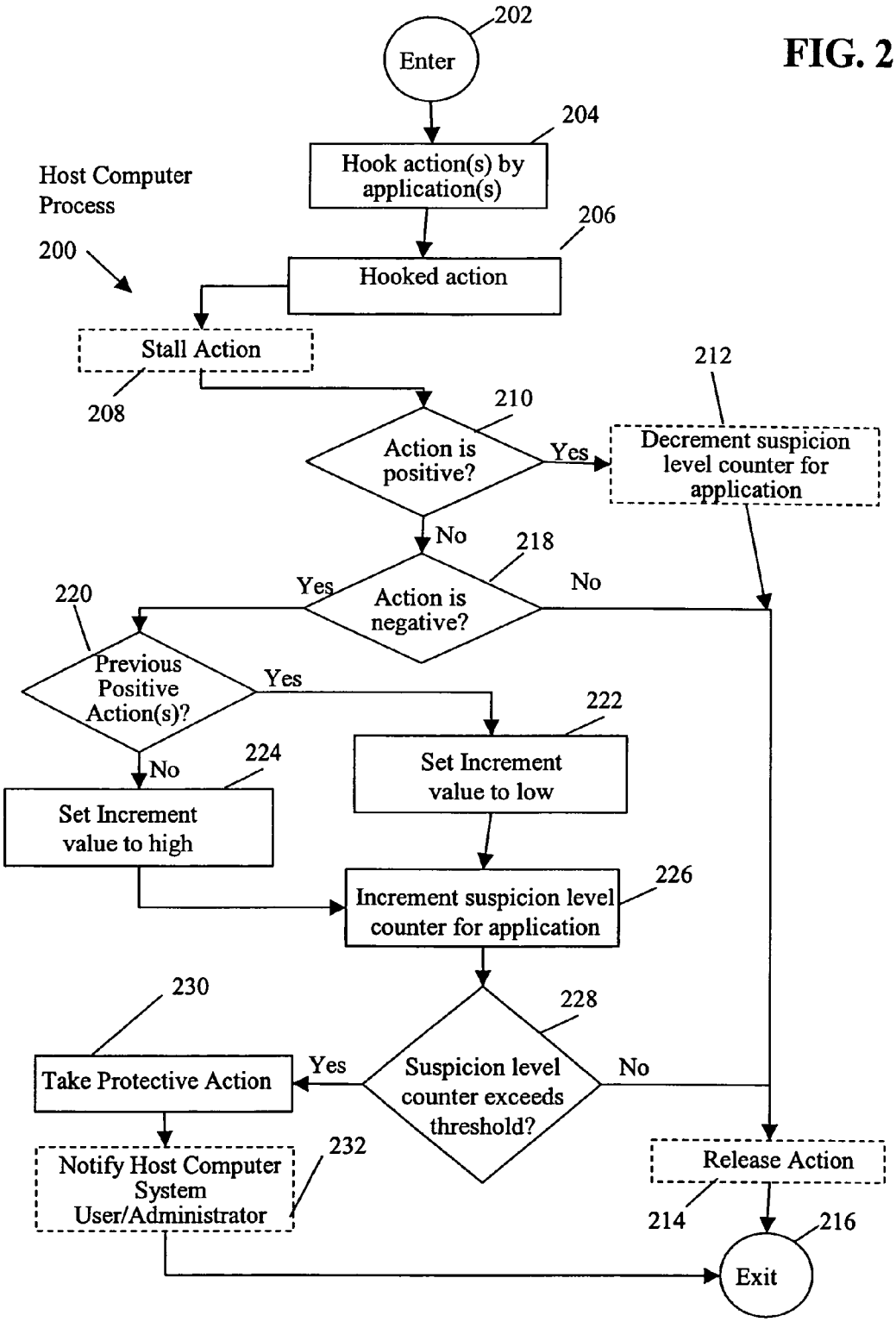


FIG. 2



BEHAVIOR BLOCKING UTILIZING POSITIVE BEHAVIOR SYSTEM AND METHOD

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the protection of computer systems. More particularly, the present invention relates to a behavior blocking system.

2. Description of the Related Art

In the computer security domain, there were behavior blocking applications that would block a suspicious action by an application on a computer system. However, a large set of these blocked suspicious actions are not malicious, i.e., are false positives.

Typically, the user of the computer system is notified that the suspicious action has been blocked and the user is required to select how the blocked suspicious action should be handled, e.g., blocked, released, blocked in the future or released in the future. Thus, these false positives are intrusive and annoying to the user of the computer system at a minimum and result in lost productivity due to the time spent by the user in responding to the false positives.

SUMMARY OF THE INVENTION

A method includes decreasing a suspicion of a negative action by an application if the application has previously performed a positive action. The positive action is an action that is never or rarely taken by malicious code. In one embodiment, the positive action is use of a user interface element by the application to have a user interaction with a user of a computer system. By taking into consideration the positive action by the application, the occurrence of false positives is minimized.

Embodiments in accordance with the present invention are best understood by reference to the following detailed description when read in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a diagram of a client-server system that includes a behavior blocking application executing on a host computer system in accordance with one embodiment of the present invention; and

FIG. **2** is a flow diagram of a host computer process in accordance with one embodiment of the present invention. Common reference numerals are used throughout the drawings and detailed description to indicate like elements.

DETAILED DESCRIPTION

In accordance with one embodiment, referring to FIG. **2**, a method includes determining that an action by an application is negative (check operation **218**). Upon a determination that the action is negative, the method includes determining if the application has had at least one positive action prior to the negative action (check operation **220**). A suspicion level counter for the application is incremented more (operations **224**, **226**) or less (operations **222**, **226**) depending upon whether the application has had at least one positive action prior to the negative action. By taking into consideration the

More particularly, FIG. **1** is a diagram of a client-server system **100** that includes a behavior blocking application **106** executing on a host computer system **102**, e.g., a first computer system, in accordance with one embodiment of the present invention.

Host computer system **102**, sometimes called a client or user device, typically includes a central processing unit (CPU) **108**, hereinafter processor **108**, an input output (I/O) interface **110**, and a memory **114**.

Host computer system **102** may further include standard devices like a keyboard **116**, a mouse **118**, a printer **120**, and a display device **122**, as well as, one or more standard input/output (I/O) devices **123**, such as a compact disk (CD) or DVD drive, floppy disk drive, or other digital or waveform port for inputting data to and outputting data from host computer system **102**. In one embodiment, behavior blocking application **106** is loaded into host computer system **102** via I/O device **123**, such as from a CD, DVD or floppy disk containing behavior blocking application **106**.

Host computer system **102** is coupled to a server system **130** of client-server system **100** by a network **124**. Server system **130** typically includes a display device **132**, a processor **134**, a memory **136**, and a network interface **138**.

Further, host computer system **102** is also coupled to a hacker computer system **104** of client-server system **100** by network **124**. In one embodiment, hacker computer system **104** is similar to host computer system **102**, for example, includes a central processing unit, an input output (I/O) interface, and a memory. Hacker computer system **104** may further include standard devices like a keyboard, a mouse, a printer, a display device and an I/O device(s). The various hardware components of hacker computer system **104** are not illustrated to avoid detracting from the principles of the invention.

Network **124** can be any network or network system that is of interest to a user. In various embodiments, network interface **138** and I/O interface **110** include analog modems, digital modems, or a network interface card.

Behavior blocking application **106** is stored in memory **114** of host computer system **102** and executed on host computer system **102**. The particular type of and configuration of host computer system **102**, hacker computer system **104**, and server system **130** are not essential to this embodiment of the present invention.

FIG. **2** is a flow diagram of a host computer process **200** in accordance with one embodiment of the present invention. Referring now to FIGS. **1** and **2** together, execution of behavior blocking application **106** by processor **108** results in the operations of host computer process **200** as described below in one embodiment.

From an enter operation **202**, flow moves to a hook action(s) by application(s) operation **204**. In hook action(s) by application(s) operation **204**, one or more applications executed on host computer system **102** are hooked. Generally, an application is hooked by hooking and intercepting specific action(s), sometimes called hooked action(s), of the application.

More particularly, in hook action(s) by application(s) operation **204**, one or more actions of one or more applications are hooked. To illustrate, a file system filter driver in the Windows operating system hooks file events by installing a layer between the user and file system for the file events and intercepts the file events between the user and file system.

In accordance with one embodiment, an application is hooked by installing one or more user mode hooks to inter-

with the user are thus hooked actions. Hooking of applications and actions is well known to those of skill in the art and typically depends upon the particular operating system of host computer system **102**. The particular hooking technique used is not essential to the present invention.

From hook action(s) by application(s) operation **204**, flow moves to a hooked action operation **206**. In hooked action operation **206**, a hooked action, i.e., an action hooked in hook action(s) by application(s) operation **204**, is made by a hooked application. The hooked action is sometimes herein referred to as "the action" or "the action by the hooked application" for simplicity of discussion.

From hooked action operation **206**, flow moves, optionally, to a stall action operation **208** (or directly to an action is positive check operation **210** if stall action operation **208** is not performed).

In stall action operation **208**, the action by the hooked application is stalled, i.e., is prevented from being executed or otherwise implemented. From stall action operation **208**, flow moves to action is positive check operation **210**.

In action is positive check operation **210**, a determination is made as to whether the action by the hooked application is positive, i.e., is a positive action. Generally, a positive action is an action that is rarely or never performed by malicious code. In one embodiment, malicious code is defined as any computer program, module, set of modules, or code that enters a computer system without an authorized user's knowledge and/or without an authorized user's consent.

For example, malicious code rarely if ever interacts with the user, e.g., a human, of host computer system **102**. As an illustration, malicious code has no user interaction about 95% of the time and about 5% of the time uses a message box to have a very minimal user interaction. Accordingly, in one embodiment, a positive action by an application occurs when the application interacts with the user of host computer system **102**, i.e., has a user interaction. Because use of a message box is a very minimal user interaction, in one embodiment, use of a message box is not defined as a positive action although use of a message box can be a positive action if desired to be defined as such.

For example, a positive action by an application occurs when the application uses a user interface element to have a user interaction with the user.

Examples of user interactions include interactions with the user in setting up the application or using the application. For example, a user interaction occurs when the application is configured by the user. As another example, a user interaction occurs when the user selects the recipient(s) of an e-mail message or the information, e.g., attachments, to be sent with an e-mail message. Although specific examples of user interactions are provided, in light of this disclosure, it is understood that other user interactions with an application can occur, and the particular user interactions depend, for example, on the particular application.

Generally, a user interface element is an element used by a user in providing input or otherwise interacting with an application. Examples of user interface elements include: (1) check boxes; (2) radio boxes; (3) list boxes; (4) combo boxes; (5) text boxes; (6) common dialog boxes; and (7) message boxes. Although specific examples of user interface elements are provided, in light of this disclosure, it is understood that other user interface elements can be used by the user, and the particular user interface element depends, for example, on the particular application.

If a determination is made that the action by the hooked

cion level counter for application operation **212** (or directly to an optional release action operation **214** if operation **212** is not performed or directly to an exit operation **216** if operations **212** and **214** are not performed).

In one embodiment, each application has an associated suspicion level counter, which is a measure of the suspicion associated with the application. This suspicion level counter is decremented in decrement suspicion level counter for application operation **212** thus reducing the suspicion associate with the application.

Decrement suspicion level counter for application operation **212** is optional and in one embodiment is not performed. In accordance with this embodiment, the suspicion level counter associate with the application is not decremented and the suspicion associated with the application remains unchanged.

From decrement suspicion level counter for application operation **212**, flow moves to, optionally, release action operation **214**. As discussed above, stall action operation **208** is optional. Accordingly, if stall action operation **208** is performed and the action was stalled, release action operation **214** is performed to release the action.
Conversely, if stall action operation **208** was not performed, release action operation **214** is unnecessary and thus not performed.

From release action operation **214** (or directly from decrement suspicion level counter for application operation **212** if operation **214** is not performed), flow moves to and exits at exit operation **216** or returns to hooked action operation **206**.

Returning again to action is positive check operation **210**, if a determination is made that the action is not a positive action, flow moves to an action is negative check operation **218**.

In action is negative check operation **218**, a determination is made as to whether the action by the hooked application is negative, i.e., is a negative action. Generally, a negative action is an action that is highly suspicious or suggestive of malicious code.

Examples of negative actions include: (1) attacking security software; (2) sending of executable attachments; (3) copying of an application across a network; and (4) sending executable instant messengering attachments. Although specific examples of negative actions are provided, in light of this disclosure, it is understood that other negative actions can occur, and the particular negative actions depend, for example, on the particular application.

If a determination is made that the action by the hooked application is not a negative action in action is negative check operation **218**, flow moves to optional release action operation **214** (or directly to exit operation **216** if operation **214** is not performed).

Conversely, if a determination is made that the action by the hooked application is a negative action in action is negative check operation **218**, flow moves to a previous positive action(s) check operation **220**. In previous positive action(s) check operation **220**, a determination is made as to whether the hooked application has performed any positive actions prior to the present negative action.

If a determination is made that the hooked application has performed at least one positive action prior to the present negative action, flow moves to a set increment value to low operation **222**.

In set increment value to low operation **222**, the increment value for the suspicion level counter for the application is set to low. Stated another way, in set increment value to low

DOCKET ALARM

Explore Litigation Insights

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

Real-Time Litigation Alerts

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

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

Advanced Docket Research

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

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

Analytics At Your Fingertips

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

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

API

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

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

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

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

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