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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 July 2008 (10.07.2008)

PCT

(10) International Publication Number
WO 2008/083382 A1

(51) International Patent Classification:
G06F 21/00 (2006.01) *G06F 15/00* (2006.01)

(21) International Application Number:
PCT/US2007/089221

(22) International Filing Date:
31 December 2007 (31.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/618,470 29 December 2006 (29.12.2006) US

(71) Applicant (*for all designated States except US*): MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: ROGERS, Justin; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). LAWRENCE, Eric M.; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). BRIDGE, Henry F.; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

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

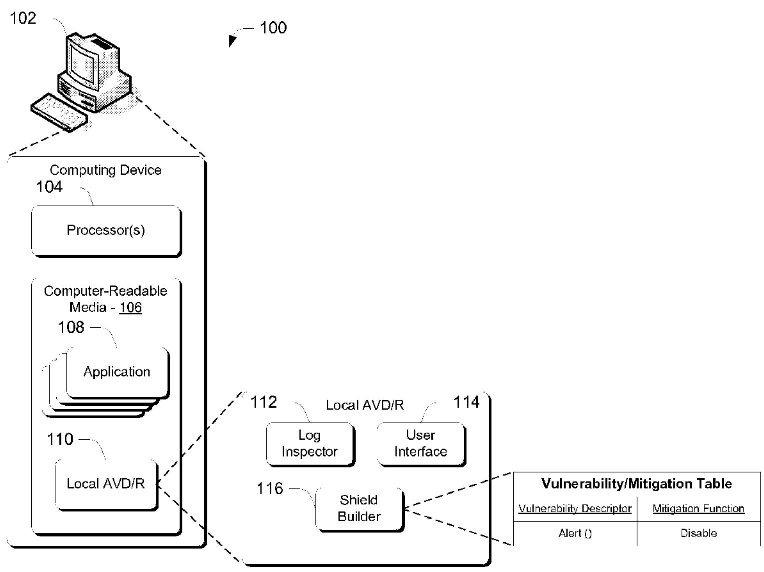
(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, 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).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:
— *with international search report*

[Continued on next page]

(54) Title: AUTOMATIC VULNERABILITY DETECTION AND RESPONSE



(57) Abstract: Various embodiments detect security vulnerabilities and, responsively, can modify an affected program so that even if an exploit runs, the program's integrity can be maintained. In at least some embodiments, a local automatic vulnerability detection and response (AVD/R) component executes on a user's local machine to detect and mitigate potential vulnerabilities through the use of a shield; and, a remote automatic vulnerability detection and response (AVD/R) component executes to report perceived vulnerabilities so that one or more shields can be delivered and applied locally to mitigate perceived vulnerabilities.

WO 2008/083382 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

Automatic Vulnerability Detection and Response

BACKGROUND

Many methods of mitigating software security vulnerabilities are reactive and quite time intensive. That is, once a vulnerability is discovered, software companies typically release a patch some time later directed to preventing attackers from exploiting the vulnerability. While this strategy has worked well for protecting users in the past, its effectiveness partially requires (1) vulnerability finders to find vulnerabilities before hackers, (2) vulnerability finders to report problems to software companies before they disclose them publicly, and (3) high patch adoption rates, so that if an exploit is developed, adopting users are protected from it.

Unfortunately, recent trends do not bode well for these requirements. Specifically, the rate of 0-day exploits (i.e., exploits that have been released for undisclosed, unfixed security vulnerabilities) has increased, and patch adoption rates continue to be slow. In order to prevent the security landscape from worsening considerably, software makers must find a way to both discover and mitigate vulnerabilities faster.

SUMMARY

Various embodiments detect security vulnerabilities and, responsively, can modify an affected program so that even if an exploit runs, the program's integrity can be maintained.

In at least some embodiments, a local automatic vulnerability detection and response (AVD/R) component executes on a user's local machine to detect and mitigate potential vulnerabilities through the use of a shield; and, a remote automatic

vulnerability detection and response (AVD/R) component executes to report perceived vulnerabilities so that one or more shields can be delivered and applied locally to mitigate perceived vulnerabilities.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a system in accordance with one embodiment.

Fig. 2 is a flow diagram that describes steps in a method in accordance with one embodiment.

Fig. 3 illustrates a system in accordance with one embodiment.

Fig. 4 is a flow diagram that describes steps in a method in accordance with one embodiment.

Fig. 5 illustrates a system in accordance with one embodiment.

Fig. 6 is a flow diagram that describes steps in a method in accordance with one embodiment.

DETAILED DESCRIPTION

Overview

Various embodiments detect security vulnerabilities and, responsively, can modify an affected program so that even if an exploit runs, the program's integrity can be maintained.

In at least some embodiments, a local automatic vulnerability detection and response (AVD/R) component executes on a user's local machine to detect and mitigate potential vulnerabilities through the use of a shield; and, a remote automatic vulnerability detection and response (AVD/R) component executes to report perceived vulnerabilities so that one or more shields can be delivered and applied locally to mitigate perceived vulnerabilities.

In the discussion that follows, a section entitled “Security Vulnerabilities In General” is provided and describes, very generally, the notion of a security vulnerability and how it can arise. Following this, a section entitled “Local AVD/R” is provided and discusses local solutions to vulnerability detection and response. Following this, a section entitled “Remote AVD/R” is provided and discusses various remote solutions to vulnerability detection. Finally, a section entitled “Using Both Local and Remote AVD/R” is provided and describes how both the local and remote approaches can be applied to provide a continuum of protection.

Security Vulnerabilities In General

Many security vulnerabilities emanate from programming errors. There are many different types of programming errors that can lead to a vulnerability. For example, a common programming error is one that allows for a buffer overflow. In situations like this, a programmer may have allocated a certain amount of space to hold data. An exploiter might figure out that if you provide the program with more data than it expects, and if the programmer did not put the correct checks in place to mitigate or eliminate the possibilities of a buffer overflow, then this excess data can cause an overflow. Using the overflow condition, the exploiter can append data or code to the end of the data received in the buffer and cause an overflow. If the data or code that is appended is executed, it can change the program or modify its functionality in some way. Hence, by virtue of a programming error, a security vulnerability can be exploited.

Often, however, exploitations of a security vulnerability can lead to program crashes. In the example above, the exploitation may make the program look to some random part of memory and start to run code that causes an invalid operation and hence, causes the program to crash.

Hence, from a program crash, one can infer that there is a problem with the program. This problem may be associated with a security vulnerability. That is, program crashes are often a sign of a vulnerability because: (1) many of the same programming errors that cause program crashes are exploitable; (2) exploit development involves a good amount of trial and error--consequently, during the early stages of exploit development, failed attempts will cause the program to crash; and (3) exploits often only work on specific versions of a program, and will sometimes crash other versions.

Local AVD/R

Fig. 1 illustrates a system in accordance with one embodiment, generally at 100. System 100 includes a computing device 102 having one or more processors 104, one or more computer-readable media 106 and one or more applications 108 that reside on the computer-readable media and which are executable by the processor(s). In addition, computing device 102 includes a local AVD/R component 110 which, in this example, is implemented in software.

Although computing device 102 is illustrated in the form of a desktop computer, it is to be appreciated and understood that other computing devices can be utilized without departing from the spirit and scope of the claimed subject matter. For example, other computing devices can include, by way of example and not limitation, portable computers, handheld computers such as personal digital assistants (PDAs), cell phones and the like.

In this example, local AVD/R component 110 includes a log inspector 112, a user interface component 114 and a shield builder 116. In operation, when a program on the local computing device crashes, information associated with the crash is entered into a crash log, as will be appreciated by the skilled artisan. Typically a crash log contains information that describes parameters associated with a particular crash. For example, the crash log can contain information that describes the program that crashed, which function or interface within the program crashed, and/or any parameters associated with the function or interface that caused the crash. Local AVD/R component 110's log inspector 112 can monitor for crashes and, when one occurs, can automatically inspect the crash log for information associated with the crash. This can include ascertaining which function or interface is associated with the crash. Once the log inspector has ascertained the cause of the crash, it can employ shield builder 116 to build a shield that effectively provides an automatic runtime solution which disables the function or interface. This fact can then be reported to the user via user interface 114.

As an example, consider the following. Assume that the user is using their browser application and a function Alert () crashes. Responsive to the crash, the crash log is updated with information that pertains to the crash, such as the name of the function that crashed and where it crashed. Using this information, log inspector 112 can employ shield builder 116 to build a shield that automatically disables the Alert () function. In one or more embodiments, the shield builder can maintain a vulnerability/mitigation table, such as the one shown in the figure. Here, the vulnerability/mitigation table includes a column that lists vulnerability descriptors and a column that lists mitigation functions. The vulnerability descriptors describe the particular function or interface that is the subject of a mitigation function. The mitigation functions describe the particular mitigation functions that are being employed. In the example above, when a crash occurs, the shield builder makes an entry in the vulnerability/mitigation table and adds "Alert ()" in the vulnerability descriptor column. In addition, the shield builder adds "Disable" to the corresponding row in the mitigation function column. This tells the application—in this case, the user's browser—that the Alert () function has been disabled.

In addition, in at least some embodiments, this fact is reported to the user via the user interface component 114. Using the corresponding user interface, the user can effectively select to turn this function back on. Hence, in this embodiment, the potential presence of a vulnerability is detected and the corresponding function or interface is selectively disabled thus preventing future exploits.

Fig. 2 is a flow diagram that describes steps in a method in accordance with one embodiment. The method can be implemented in connection with any suitable hardware, software, firmware or combination thereof. In one or more embodiments, the method can be implemented in connection with a system, such as the one shown and described in Fig. 1. Other systems can be utilized without departing from the spirit and scope of the claimed subject matter.

Step 200 detects a local program crash. Examples of how this can be done are given above. Step 202 inspects a crash log to ascertain circumstances surrounding the crash. Step 204 disables a function or interface that was the subject of the crash. Examples of how can be done are provided above. Step 206 notifies a user of the disabled function or interface.

Remote AVD/R

In one or more embodiments, information associated with a program crash can be used remotely. Specifically, when a crash occurs, this information can be remotely reported for further analysis. Such analysis can include, by way of example and not limitation, analyzing the source of the crash and various associated parameters, as well as evaluating such crashes across multiple users to ascertain whether there is a pattern associated with the crash. If a vulnerability is detected, a corresponding shield can be built and provided to users for protecting against exploitations that seek to exploit the vulnerability.

As an example, consider Fig. 3. There, a system is illustrated in accordance with one embodiment, generally at 300. System 300 includes a computing device 302 having one or more processors 304, one or more computer-readable media 306 and one or more applications 308 that reside on the computer-readable media and which are executable by the processor(s). In addition, computing device 302 includes a remote AVD/R component 310 which, in this example, is implemented in software.

Although computing device 302 is illustrated in the form of a desktop computer, it is to be appreciated and understood that other computing devices can be utilized without departing from the spirit and scope of the claimed subject matter. For example, other computing devices can include, by way of example and not limitation, portable computers, handheld computers such as personal digital assistants (PDAs), cell phones and the like.

In this example, remote AVD/R component 310 includes a log inspector 312, a user interface component 314 and a crash reporter component 316. In operation, when a program on the local computing device crashes, information associated with the crash is entered into a crash log, as described above. Remote AVD/R component 310's log inspector 312 can monitor for crashes and, when one occurs, can automatically inspect the crash log for information associated with the crash. This can include ascertaining which function or interface is associated with the crash. Once the log inspector has ascertained the cause of the crash, the remote AVD/R component can, via user interface 314, ask the user if the user wishes to report the crash to a remote server for further analysis. If the user chooses to report the crash information, the information is aggregated and analyzed by the server. In at least some embodiments, analysis of the crash information can include employing human experts to analyze and ascertain whether any exploitations have been employed.

In an event that analysis of the crash log(s) indicates that a vulnerability has been exploited, one or more shields, such as those described above, can be developed and employed, as by being downloaded and applied locally. In one or more embodiments, user interface 314 can provide the user with an option to re-enable the function or interface that was or is disabled.

Fig. 4 is a flow diagram that describes steps in a method in accordance with one embodiment. The method can be implemented in connection with any suitable hardware, software, firmware or combination thereof. In one or more embodiments, the method can be implemented in connection with a system, such as the one shown and described in Fig. 3. Other systems can be utilized without departing from the spirit and scope of the claimed subject matter.

Step 400 detects a local program crash. Examples of how this can be done are given above. Step 402 inspects a crash log to ascertain circumstances surrounding the crash. Step 404 queries a user to ascertain whether the crash can be reported. Step 406 reports the crash to a remote server in the event the user has given authorization. Step 408 receives and implements a shield which effectively disables a function or interface that was the subject of the crash. The shield can be received by downloading it over a network, such as the Internet. As part of this step, the user interface can be utilized to give the user an option to disable the function or interface, or to re-enable the function or interface. Examples of how this can be done are provided above.

Using Both Local and Remote AVD/R

In one or more embodiments, information associated with a program crash can be used both locally and remotely. Specifically, when a crash occurs, this information can be used locally to disable the affected function or interface by applying a shield. In addition, this information can be used remotely to conduct analysis as described above. Such analysis can include, by way of example and not limitation, analyzing the source of the crash and various associated parameters, as well as evaluating such crashes across multiple users to ascertain whether there is a pattern associated with the crash. If a vulnerability is detected, a corresponding shield can be built and provided to users for protecting against any exploitations that seek to exploit the vulnerability.

As an example, consider Fig. 5. There, a system is illustrated in accordance with one embodiment, generally at 500. System 500 includes a computing device 502 having one or more processors 504, one or more computer-readable media 506 and one or more applications 508 that reside on the computer-readable media and which are executable by the processor(s). In addition, computing device 502 includes a local/remote AVD/R component 510 which, in this example, is implemented in software.

Although computing device 502 is illustrated in the form of a desktop computer, it is to be appreciated and understood that other computing devices can be utilized without departing from the spirit and scope of the claimed subject matter. For example, other computing devices can include, by way of example and not limitation, portable computers, handheld computers such as personal digital assistants (PDAs), cell phones and the like.

In this example, local/remote AVD/R component 510 includes a log inspector 512, a user interface component 514, a reporter component 516 and a shield builder 518. In operation, when a program on the local computing device crashes, information associated with the crash is entered into a crash log, as described above. Local/remote AVD/R component 510's log inspector 512 can monitor for crashes and, when one occurs, can automatically inspect the crash log for information associated with the crash. This can include ascertaining which function or interface is associated with the crash. Once the log inspector has ascertained the cause of the crash, the local/remote AVD/R component can apply a shield locally, as described above, to disable the function or interface that is associated with the crash. This can also be reported to the user via the user interface component 514 which can also allow the user to re-enable the function or interface that has been disabled.

In addition, in one or more embodiments, the local/remote AVD/R component can, via user interface 514, ask the user if the user wishes to report the crash to a remote server for further analysis. If the user chooses to report the crash information, the information is aggregated and analyzed by the server. In at least some embodiments, analysis of the crash information can include employing human experts to analyze and ascertain whether any exploitations have been employed.

In an event that analysis of the crash log(s) indicates that a vulnerability has been exploited, one or more shields, such as those described above, can be developed and employed, as by being downloaded and applied on a local machine. In one or more embodiments, user interface 514 can provide the user with an option to re-enable the function or interface that was or is disabled.

Fig. 6 is a flow diagram that describes steps in a method in accordance with one embodiment. The method can be implemented in connection with any suitable hardware, software, firmware or combination thereof. In one or more embodiments, the method can be implemented in connection with a system, such as the one shown and described in Fig. 6. Other systems can be utilized without departing from the spirit and scope of the claimed subject matter.

Step 600 detects a local program crash and step 602 asks the user for approval to report the crash to a remote server. If user approval is granted, at step 604, then step 606 ascertains the cause of the crash, reports the crash to the remote server and checks for any solutions that might be available for the crash. By reporting the crash to the remote server, analysis of the crash and for any associated patterns across multiple users can be conducted. Analysis can include both automatic, machine analysis and human analysis. Step 608 then downloads and applies any relevant shields locally and step 610 notifies the user.

If, on the other hand, approval is not granted at step 604, step 612 identifies the crashing function or interface and step 614 determines shield eligibility. If step 616 determines that there is an eligible local shield to address the problem, step 618 applies the shield as described above and step 620 notifies the user. If, on the other hand, there is no eligible shield, step 620 notifies the user.

The above-described embodiments can be implemented in connection with any suitable application, and can comprise part of the application, or a separate component that is utilized by the application. For example, in one or more embodiments, the functionality described above can be implemented as part of a web browser, instant messaging application or any other suitable application or software component or system. For example, the functionality can be implemented as part of an operating system.

In one or more embodiments, one or more so-called reputation services can be employed to further enhance security. Specifically, a reputation service or third party service can broadly monitor for security exploitations and report any perceived or actual vulnerabilities to the appropriate authorities. For example, a reputation service may detect that there is a security vulnerability associated with a particular function associated with a particular web page. Once detected, the reputation service can then report the vulnerability to the appropriate company, such as Microsoft, and/or cause shields to be selectively downloaded or otherwise made available to various users to address the perceived vulnerability.

Conclusion

Various embodiments detect security vulnerabilities and, responsively, can modify an affected program so that even if an exploit runs, the program's integrity can be maintained. In at least some embodiments, a local automatic vulnerability detection and response (AVD/R) component executes on a user's local machine to detect and mitigate potential vulnerabilities through the use of a shield; and, a remote automatic vulnerability detection and response (AVD/R) component executes to report perceived vulnerabilities so that one or more shields can be delivered and applied locally to mitigate perceived vulnerabilities.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

CLAIMS

1. A computer-implemented method comprising:

detecting a local program crash; and

responsive to said detecting, disabling a function or interface that was a subject of the program crash.

2. The method of claim 1 further comprising after said detecting and prior to said disabling, inspecting a crash log to ascertain the function or interface associated with the program crash.

3. The method of claim 1 further comprising notifying a user that a function or interface has been disabled.

4. The method of claim 1, wherein the act of disabling is performed by building and applying a shield.

5. The method of claim 4, wherein the shield provides an automatic, runtime solution.

6. The method of claim 4, wherein the acts of building and applying are performed by using a vulnerability/mitigation table which associates vulnerabilities and mitigation functions.

7. The method of claim 1, wherein said acts of detecting and disabling are performed by a web browser.

8. One or more computer-readable media having computer readable instructions thereon which, when executed, implement the method of claim 1.

9. A computing system embodying the one or more computer-readable media of claim 8.

10. A computer-implemented method comprising:

detecting a local program crash;

responsive to said detecting, querying a user to ascertain whether the program crash can be reported;

reporting the local program crash; and

responsive to said reporting, receiving and applying a shield effective to disable a function or interface that was a subject of the program crash.

11. The method of claim 10 further comprising notifying a user that a function or interface has been disabled.

12. The method of claim 10 further comprising providing the user, via a user interface, with an option to re-enable the function or interface that has been disabled.

13. The method of claim 10, wherein the act of applying is performed by using a vulnerability/mitigation table which associates vulnerabilities and mitigation functions.

14. The method of claim 10 further comprising responsive to said detecting, inspecting a crash log to ascertain the function or interface associated with the program crash.

15. The method of claim 10, wherein said acts of detecting, querying, reporting, receiving and applying are performed by a web browser.

16. One or more computer-readable media having computer readable instructions thereon which, when executed, implement the method of claim 10.

17. A computing system embodying the one or more computer-readable media of claim 16.

18. A computer-implemented method comprising:

detecting a local program crash;

asking a user for approval to report the crash to a remote server;

if user approval is granted:

reporting the crash to the remote server;

responsive to said reporting, downloading one or more shields configured to disable a function or interface associated with the crash;

applying the one or more shields to disable the function or interface;

if user approval is not granted, disabling a function or interface that was a subject of the program crash.

19. The method of claim 18 further comprising providing the user, via a user interface, with an option to re-enable the function or interface that has been disabled.

20. The method of claim 18, wherein the acts of detecting and asking are performed by a web browser.

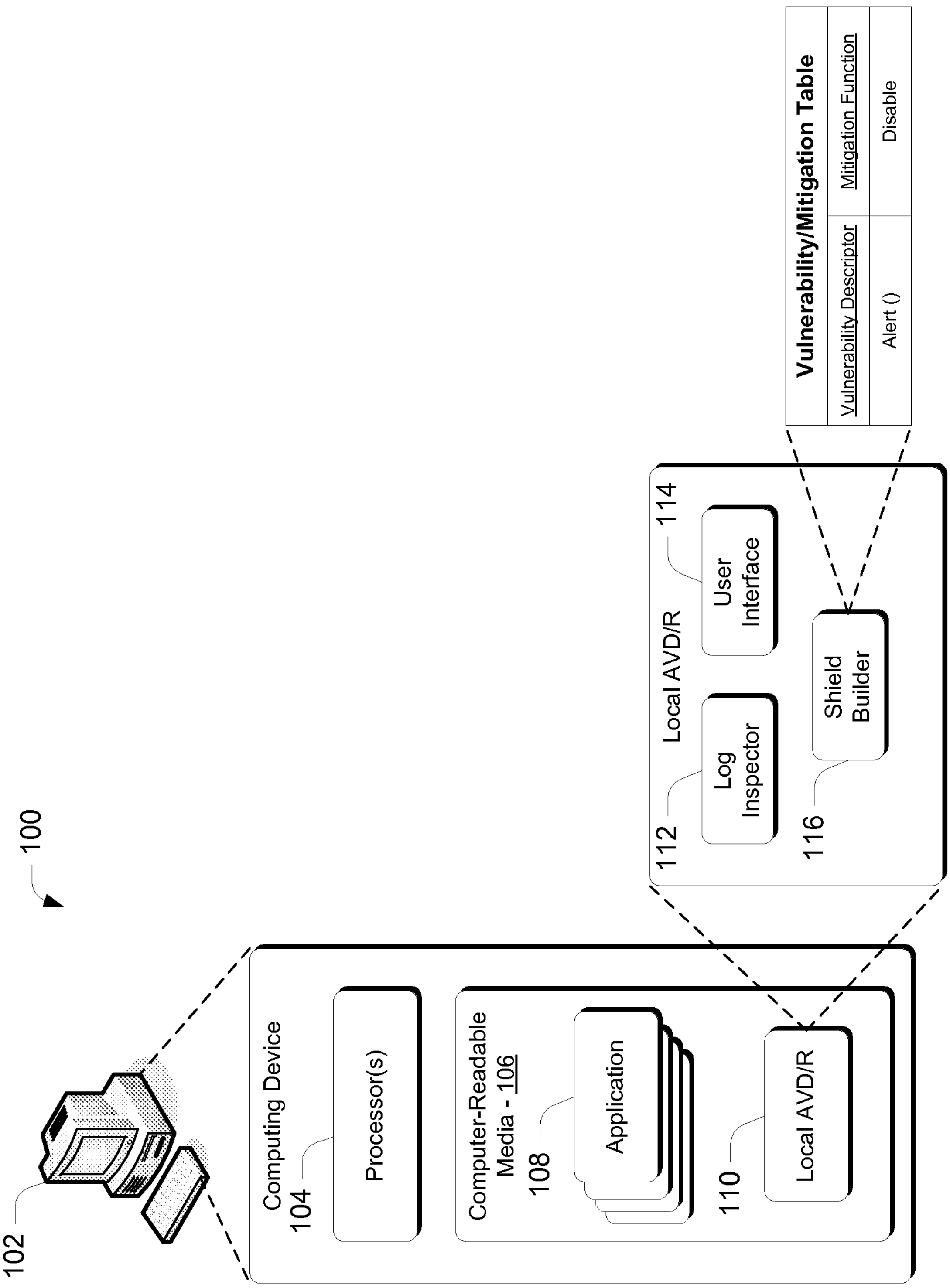

100
102
Computing Device
104
Processor(s)
Computer-Readable Media - 106
108
Application
110
Local AVD/R
Local AVD/R
112
Log Inspector
114
User Interface
116
Shield Builder
Vulnerability/Mitigation Table
Vulnerability Descriptor
Mitigation Function
Alert ()
Disable

Fig. 1

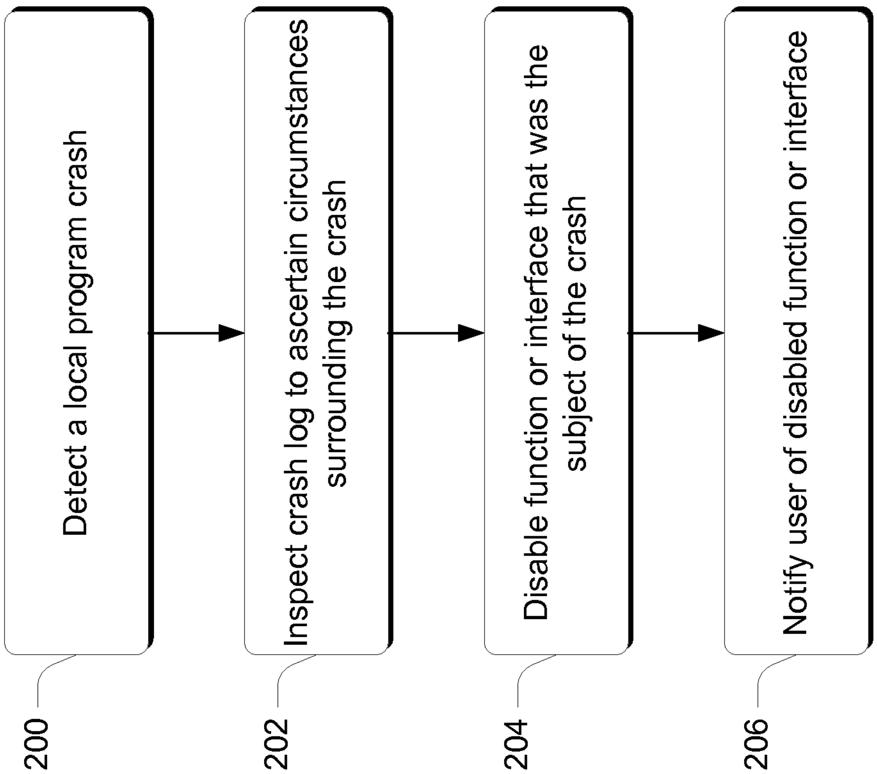

200
Detect a local program crash
202
Inspect crash log to ascertain circumstances surrounding the crash
204
Disable function or interface that was the subject of the crash
206
Notify user of disabled function or interface

Fig. 2

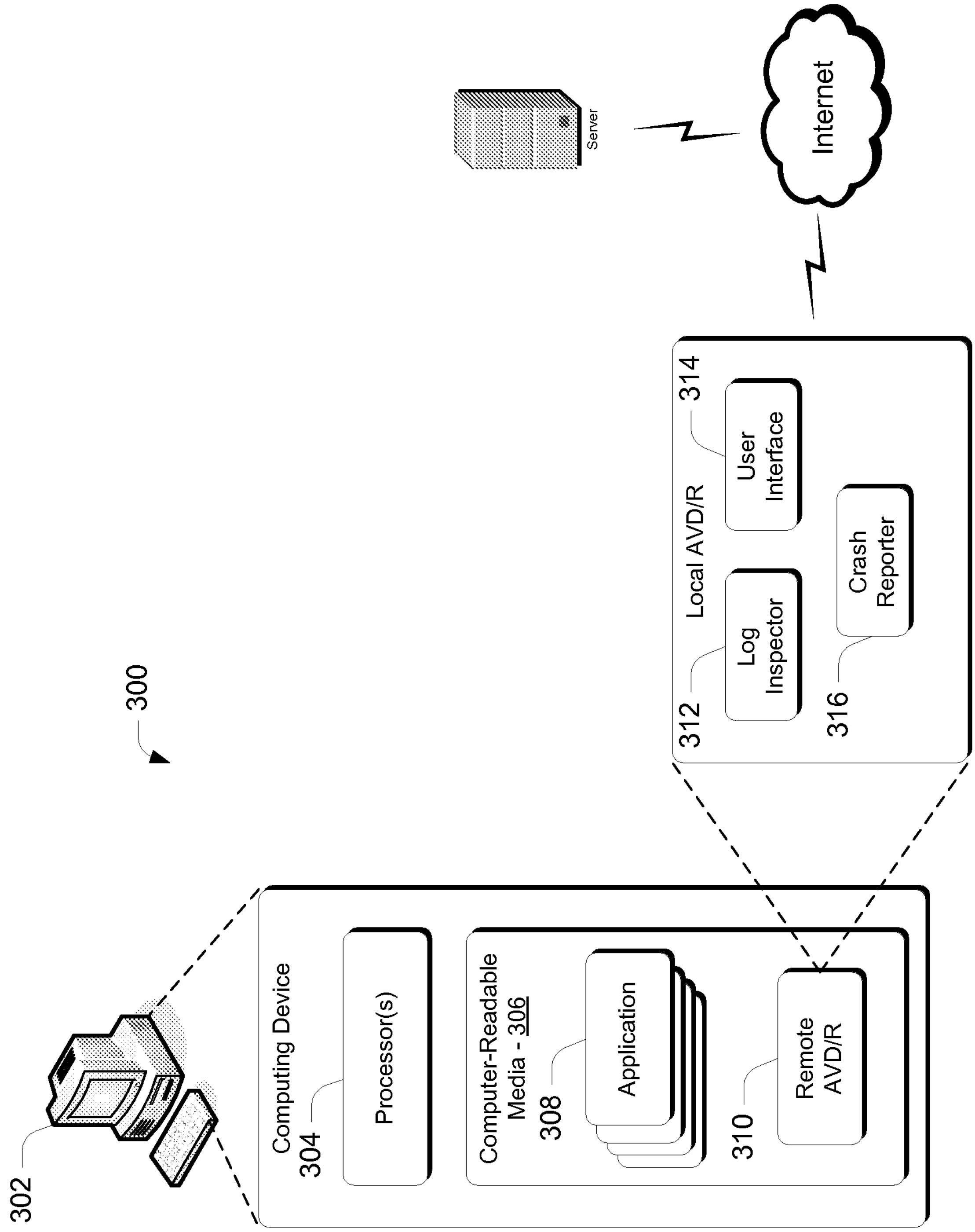
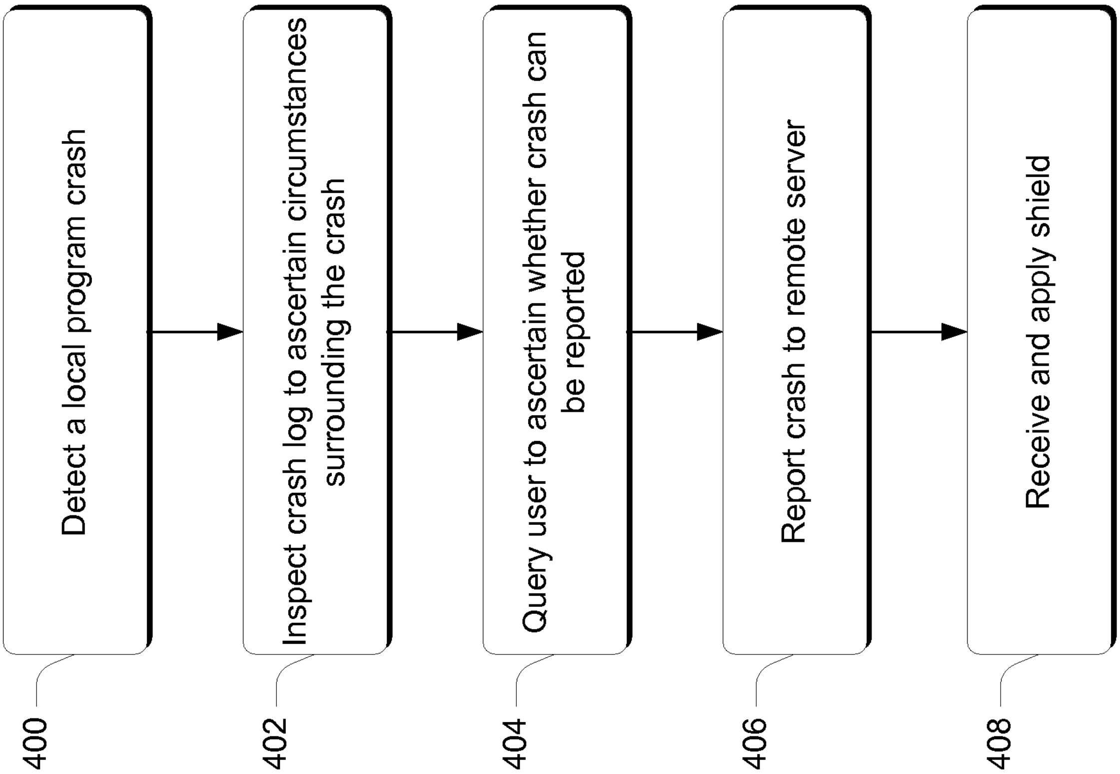

300
302
Computing Device
304
Processor(s)
Computer-Readable Media - 306
308
Application
310
Remote AVD/R
Local AVD/R
312
Log Inspector
314
User Interface
316
Crash Reporter
Server
Internet

Fig. 3

Fig. 4


Detect a local program crash
Inspect crash log to ascertain circumstances surrounding the crash
Query user to ascertain whether crash can be reported
Report crash to remote server
Receive and apply shield
400
402
404
406
408

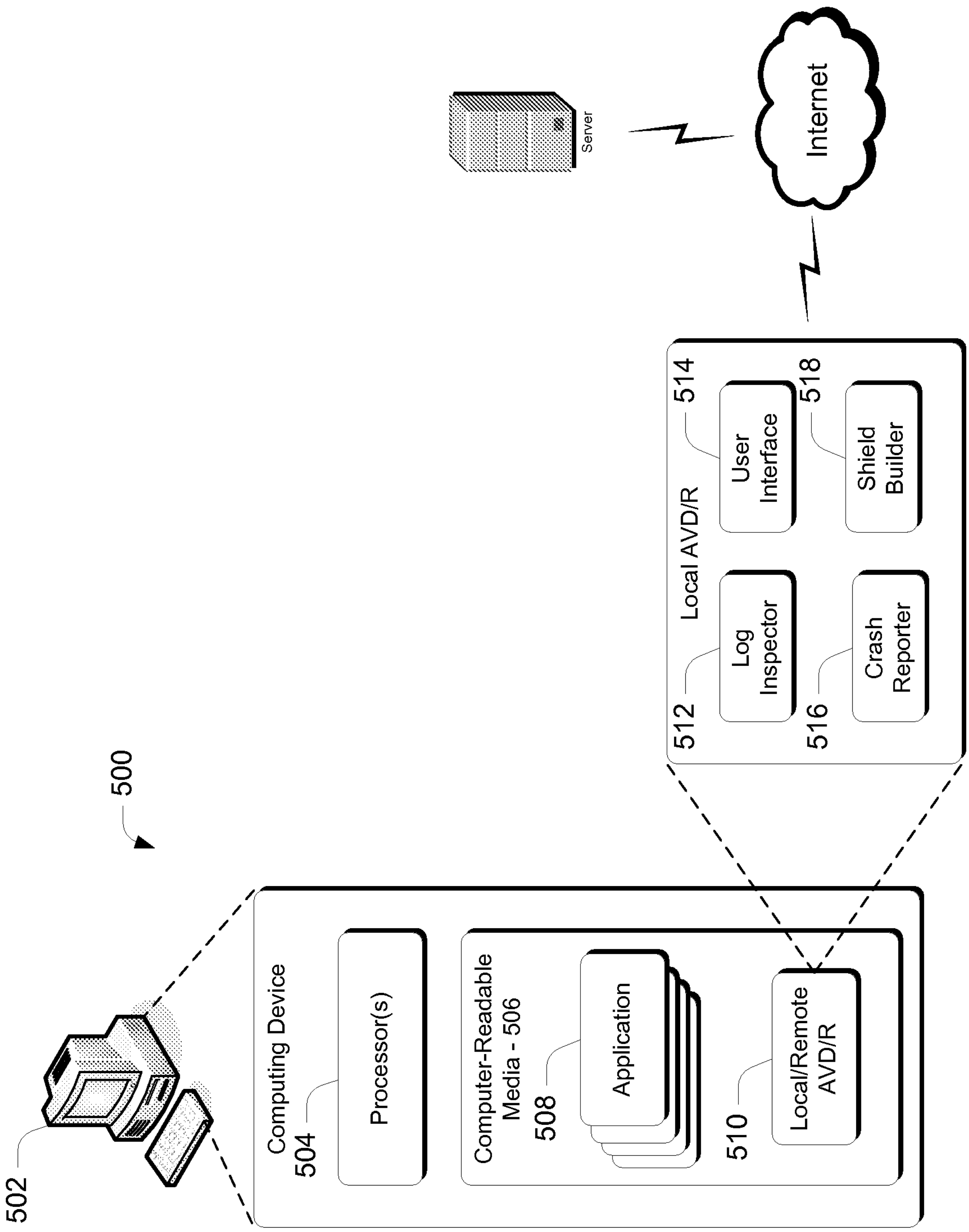

500
502
Computing Device
504
Processor(s)
Computer-Readable Media - 506
508
Application
510
Local/Remote AVD/R
Local AVD/R
512
Log Inspector
514
User Interface
516
Crash Reporter
518
Shield Builder
Internet
Server

Fig. 5

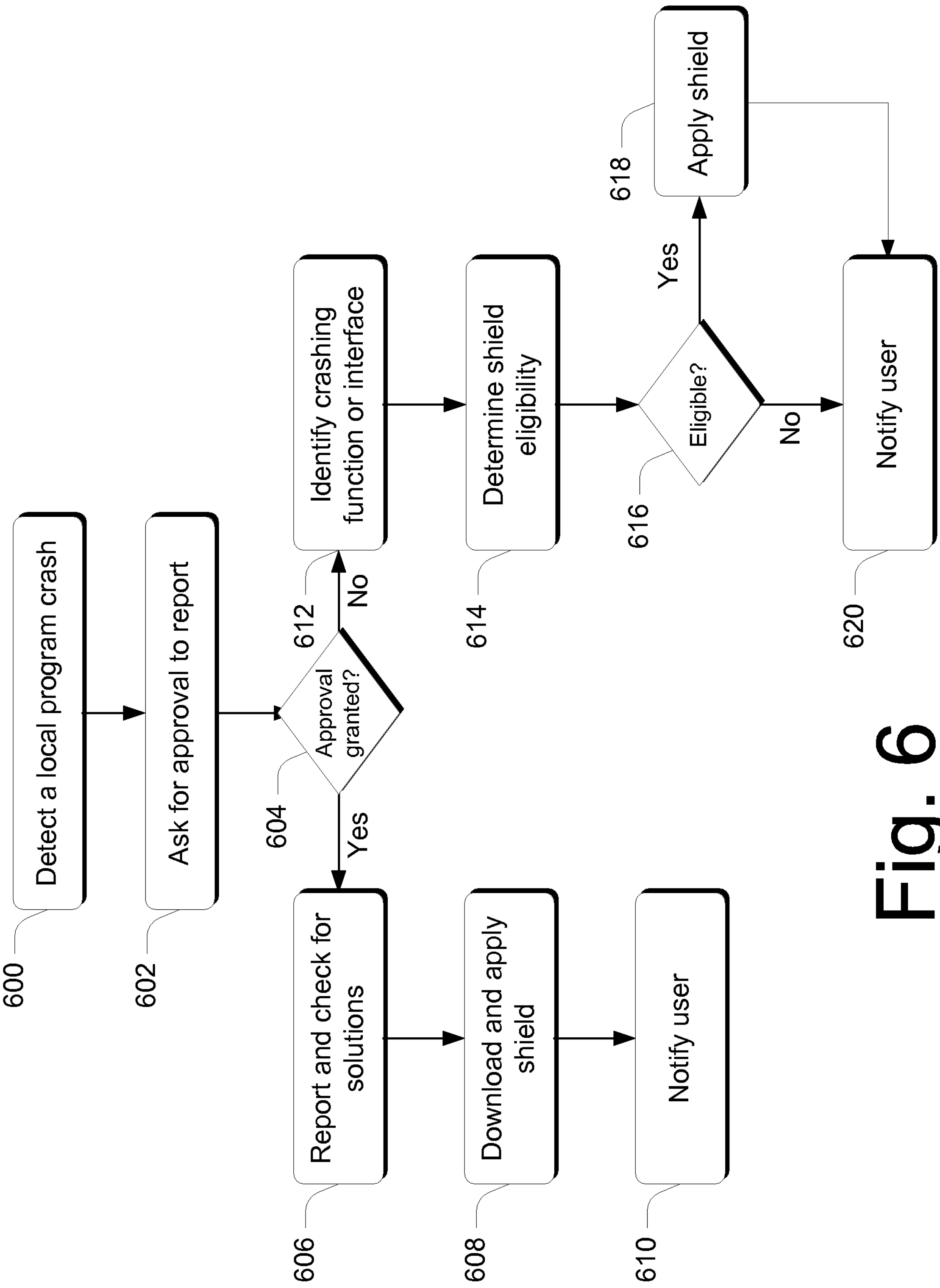

600
Detect a local program crash
602
Ask for approval to report
604
Approval granted?
Yes
No
606
Report and check for solutions
608
Download and apply shield
610
Notify user
612
Identify crashing function or interface
614
Determine shield eligibility
616
Eligible?
Yes
No
618
Apply shield
620
Notify user

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2007/089221

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/00(2006.01)i, G06F 15/00(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 8 G06F 9/42, 11/00, 15/00, 21/00, H02H 3/05

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Utility models and applications for Utility models since 1975
Japanese Utility models and applications for Utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
e-KIPASS (KIPO internal) "program","crash","function","report"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 6,629,267 B1 (Kirk A. Glerum, et al.) 30 SEPTEMBER 2003 See abstracts, claims 1 - 14, figures 2 - 3	1-3, 8-12, 14, 16-19 4-7, 13, 15, 20
A	US 2004/0205421 A1 (Kirk A. Glerum, et al.) 14 OCTOBER 2004 See abstracts, claims 12 - 14, figures 2, 7	1 - 20
A	WO 2005/024630 A1 (SCIENCE PARK CO., LTD.) 17 MARCH 2005 See abstracts, claims 1 - 8, figures 3 - 4	1 - 20
A	WO 2002/075547 A1 (KAVADO, INC.) 26 SEPTEMBER 2002 See abstracts, claims 1 - 27, figure 3	1 - 20

☐ Further documents are listed in the continuation of Box C.　　☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
02 JUNE 2008 (02.06.2008)	**02 JUNE 2008 (02.06.2008)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer
KANG, YOON SEOK
Telephone No. 82-42-481-5722

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2007/089221

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US6629267B1	30.09.2003	NONE	
US20040205421A1	14.10.2004	US7257743BB	14.08.2007
WO2005024630A1	17.03.2005	CN1886728A	27.12.2006
		EP1662379A1	31.05.2006
		KR102006056998A	25.05.2006
		US20070101317A1	03.05.2007
WO2002075547A1	26.09.2002	AU2002252371B2	03.10.2002
		CA2441230A1	26.09.2002
		EP1381949A1	21.01.2004
		JP1653676A	04.11.2004
		KR1020030096277A	24.12.2003
		NZ527915A	27.05.2005
		US07313822B2	25.12.2007
		US20030023873A1	30.01.2003
		US2003204719A1	30.10.2003