



US 20140245450A1

(19) **United States**

(12) **Patent Application Publication**
Oberheide et al.

(10) **Pub. No.: US 2014/0245450 A1**

(43) **Pub. Date: Aug. 28, 2014**

(54) **SYSTEM AND METHOD FOR PATCHING A
DEVICE THROUGH EXPLOITATION**

Publication Classification

(71) Applicant: **Duo Security, Inc.**, Ann Arbor, MI (US)

(72) Inventors: **Jon Oberheide**, Ann Arbor, MI (US);
Douglas Song, Ann Arbor, MI (US)

(73) Assignee: **Duo Security, Inc.**, Ann Arbor, MI (US)

(51) **Int. Cl.**

G06F 21/57 (2006.01)

H04L 29/06 (2006.01)

(52) **U.S. Cl.**

CPC **G06F 21/577** (2013.01); **H04L 63/1433**
(2013.01)

USPC **726/25**

(21) Appl. No.: **14/188,492**

(22) Filed: **Feb. 24, 2014**

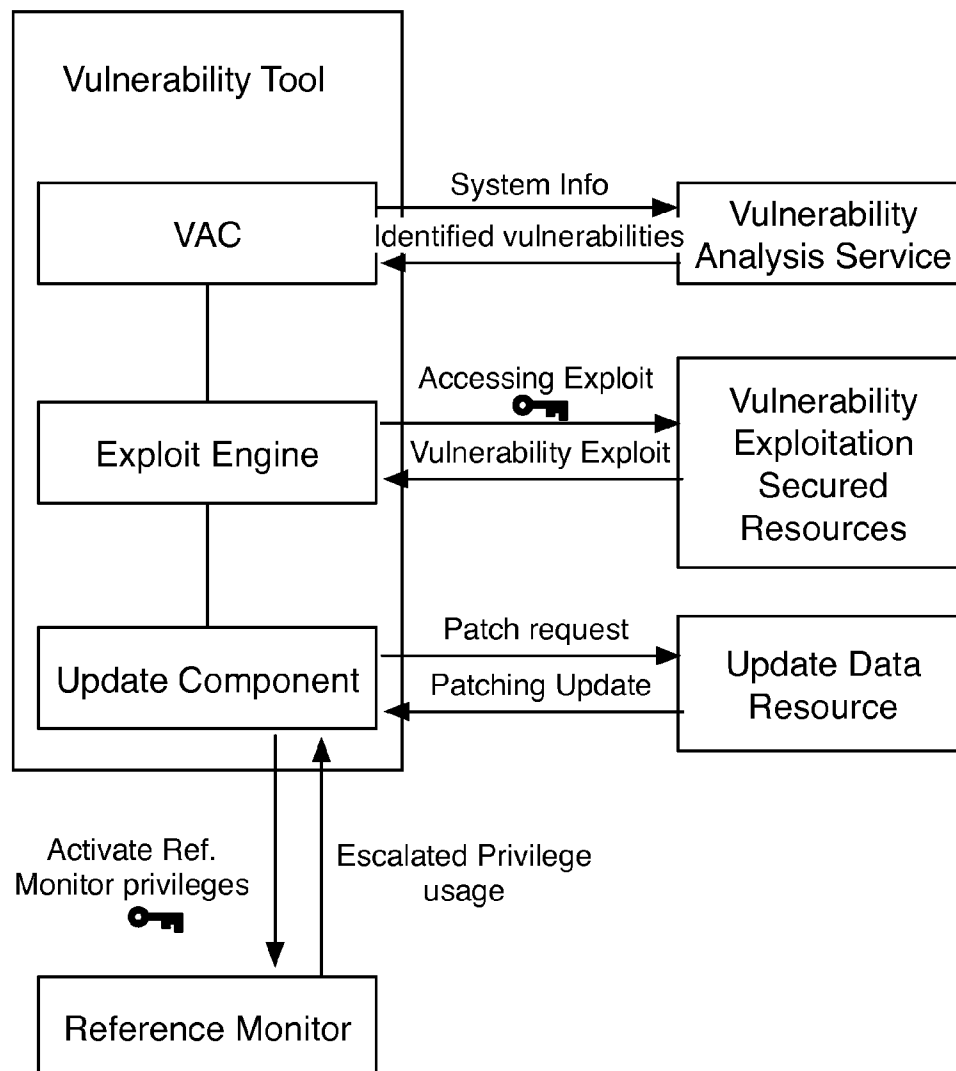
Related U.S. Application Data

(60) Provisional application No. 61/768,256, filed on Feb.
22, 2013.

(57)

ABSTRACT

A system and method that includes identifying a vulnerability in a computing device; accessing a vulnerability exploitation mapped to the identified vulnerability; at the computing device, executing the vulnerability exploitation and entering an operating mode of escalated privileges; and while in the operating mode of escalated privileges, updating the system with a vulnerability resolution.



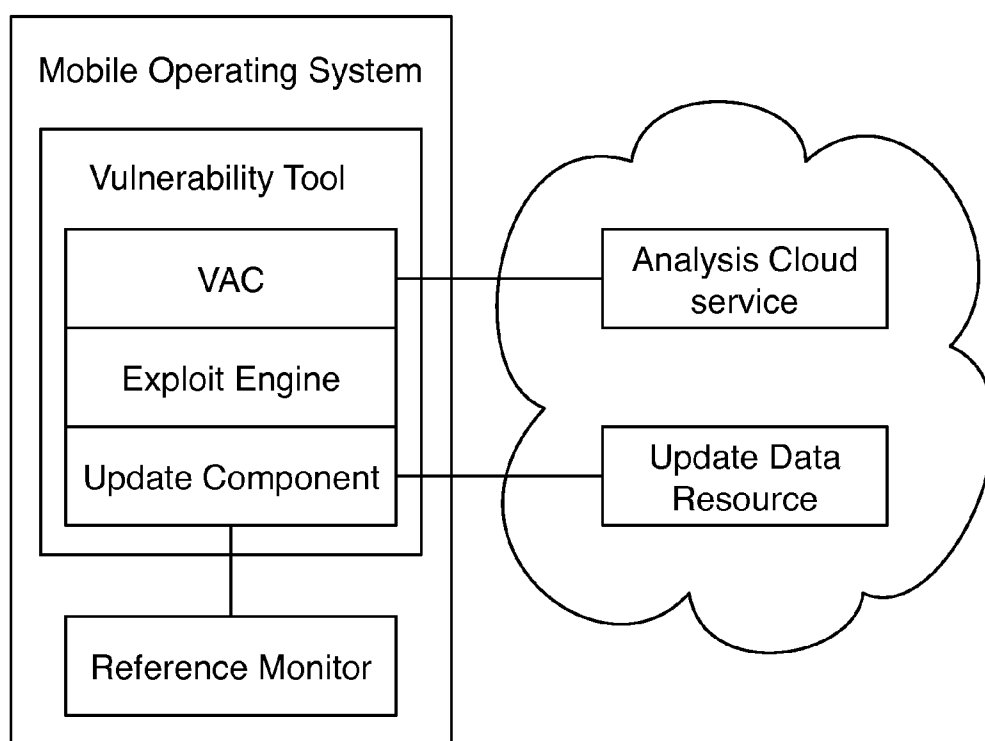


FIGURE 1

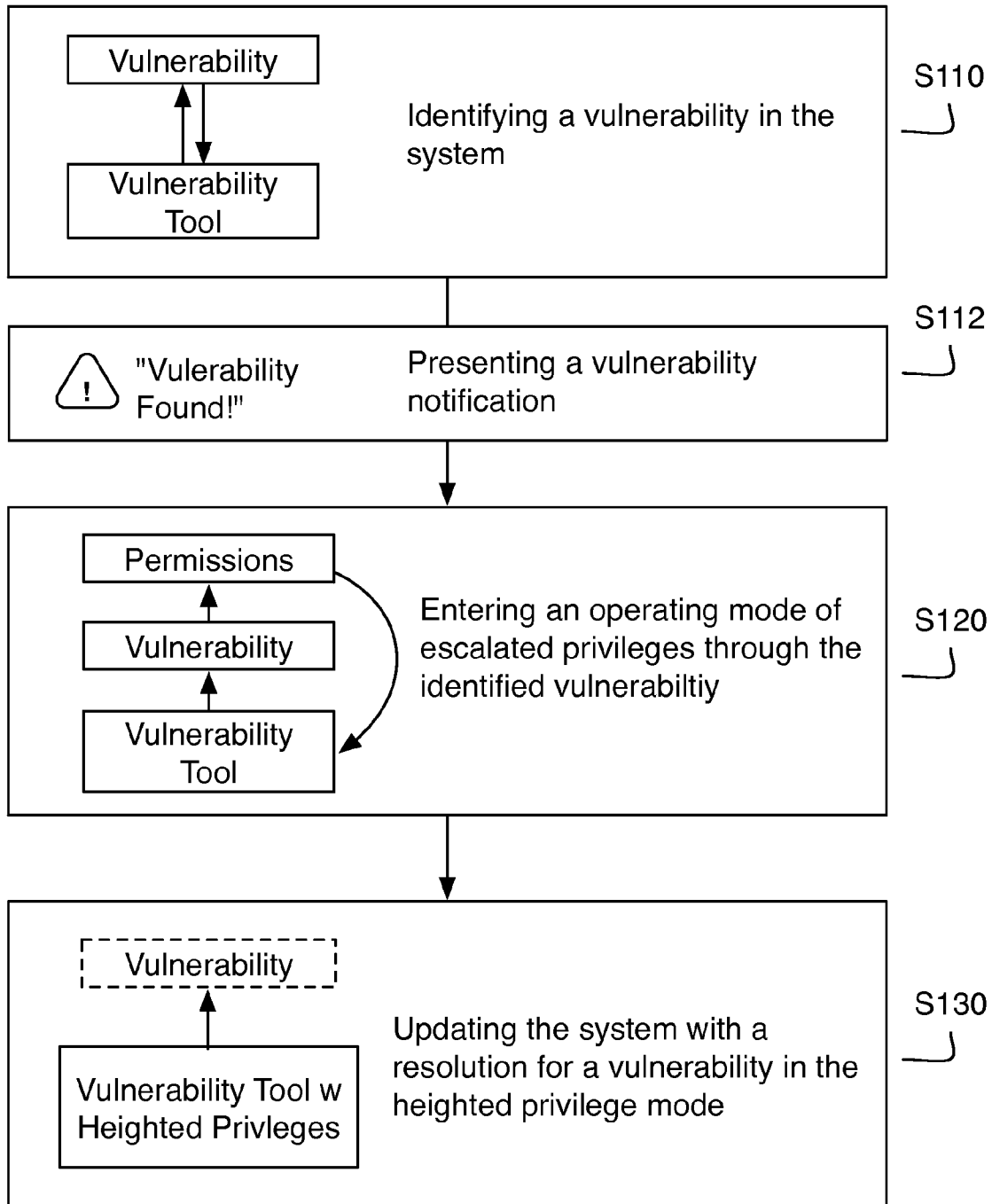


FIGURE 2

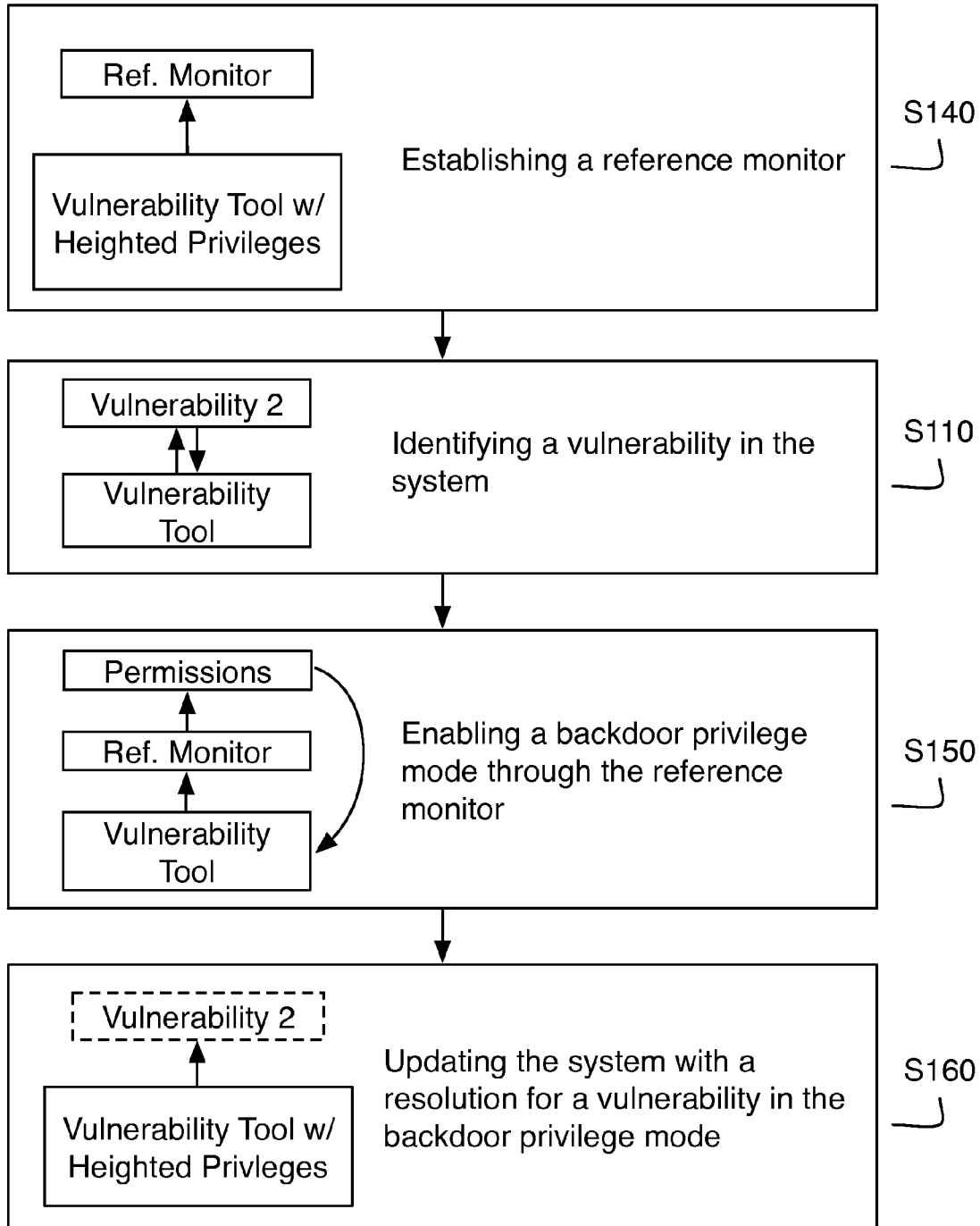


FIGURE 3

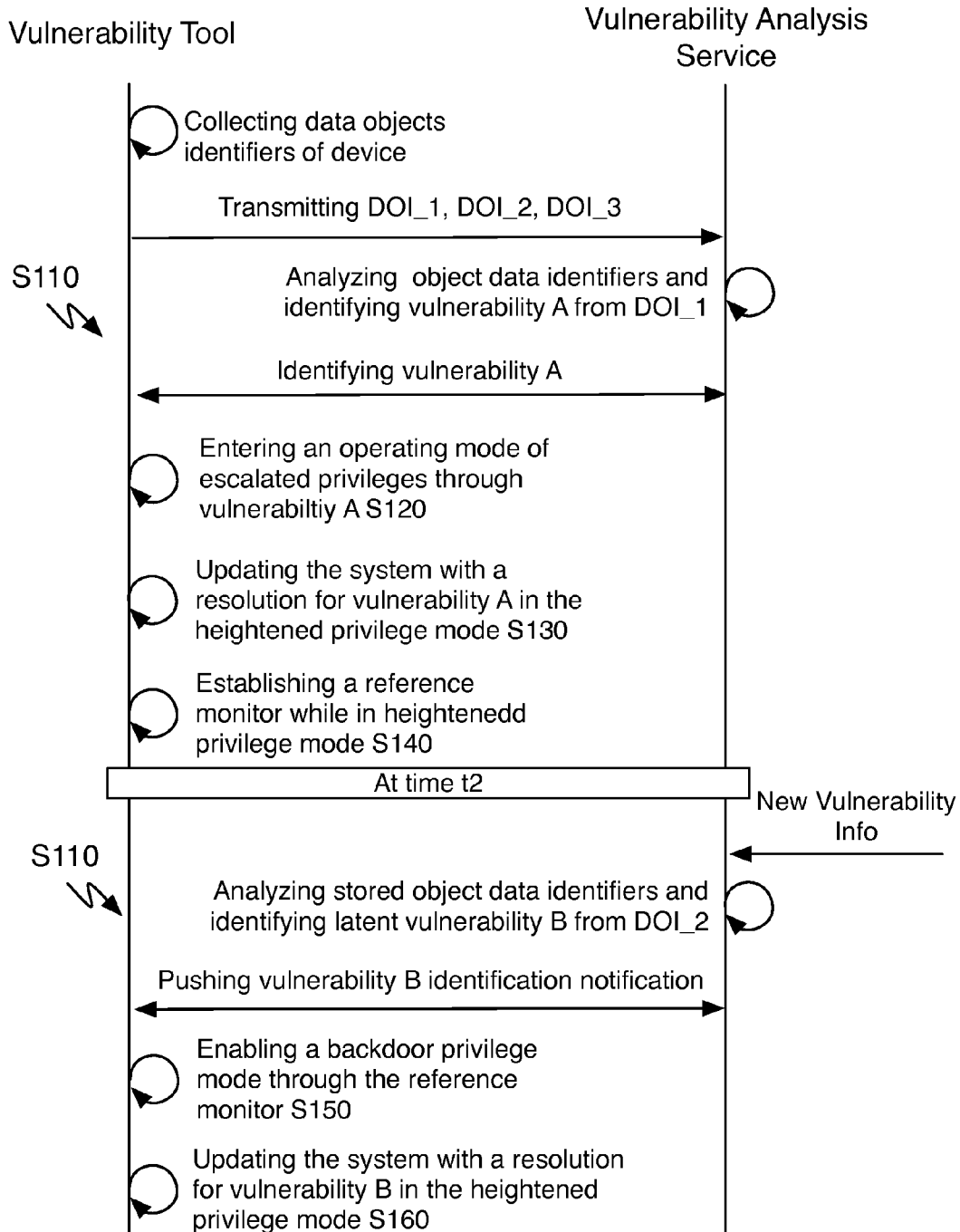


FIGURE 4

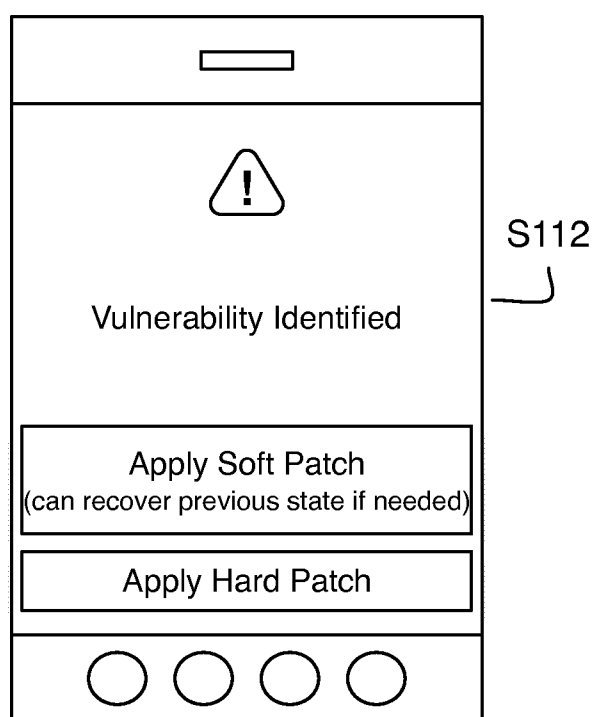


FIGURE 5

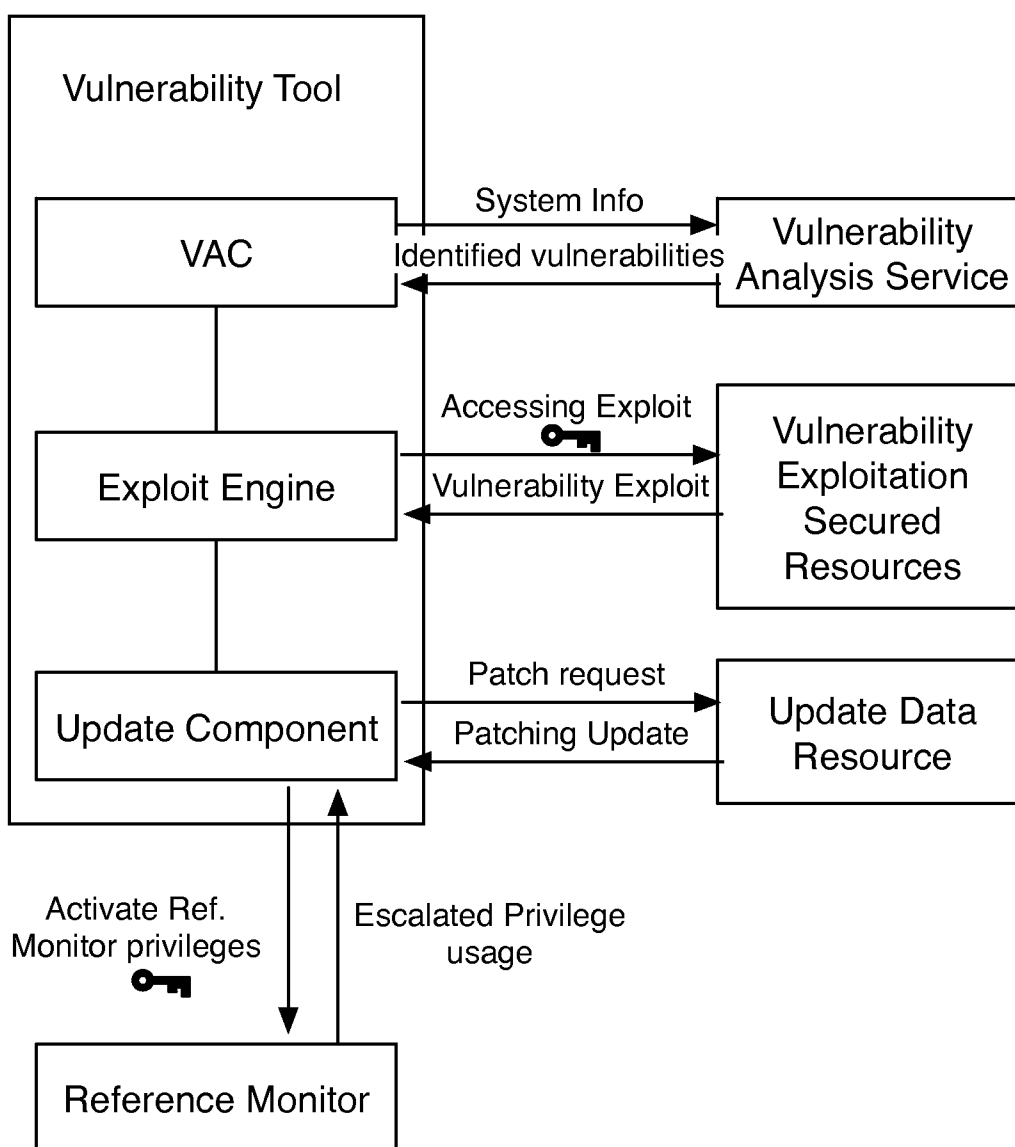


FIGURE 6

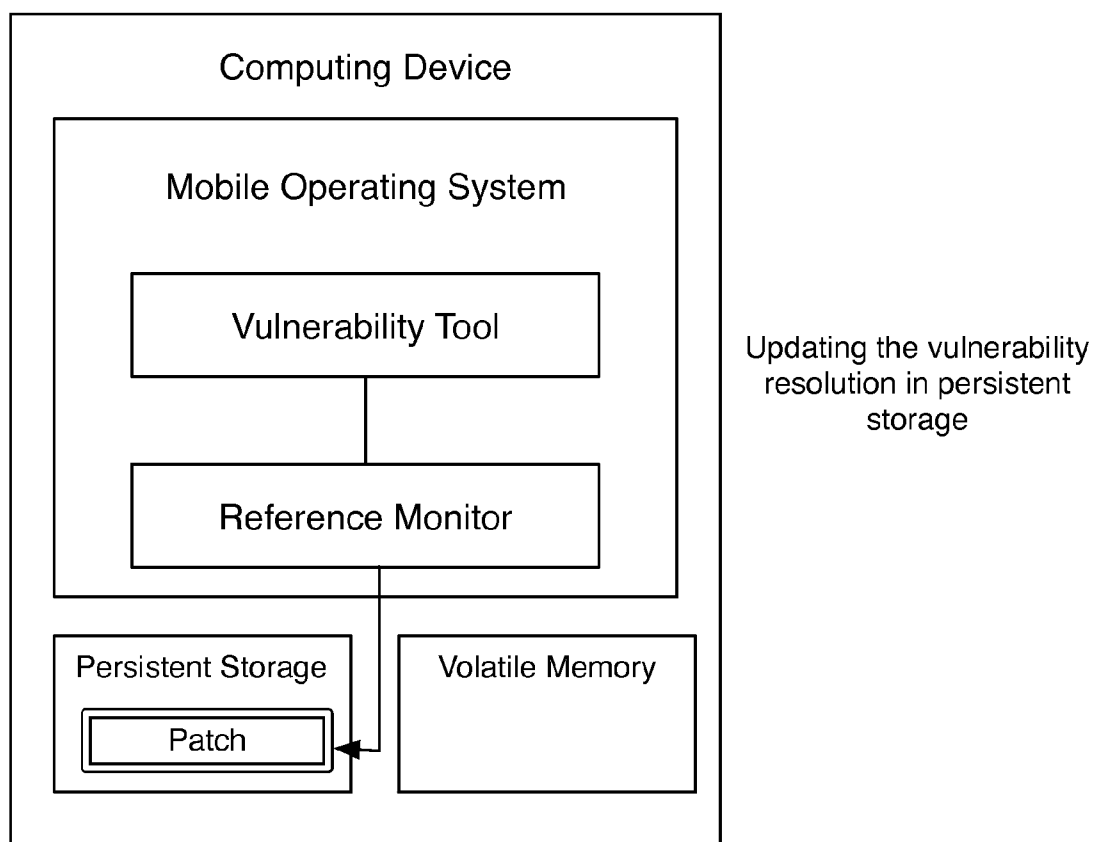


FIGURE 7

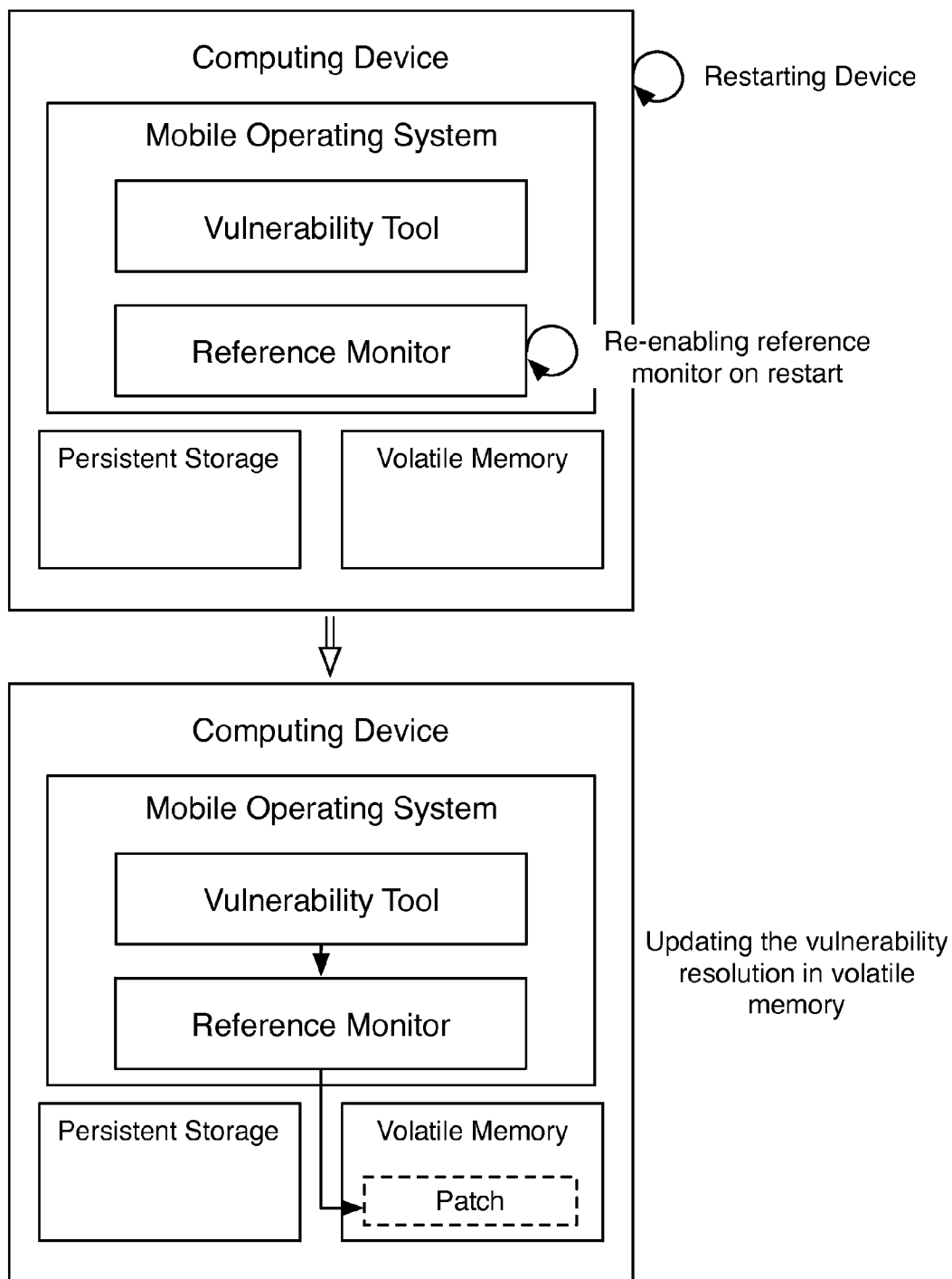


FIGURE 8

SYSTEM AND METHOD FOR PATCHING A DEVICE THROUGH EXPLOITATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application Ser. No. 61/768,256, filed on 22 Feb. 2013, which is incorporated in its entirety by this reference.

TECHNICAL FIELD

[0002] This invention relates generally to the digital device security field, and more specifically to a new and useful portable card display in the digital device security field.

BACKGROUND

[0003] Many parties control the software underlying a modern mobile device. For a mobile operating system, there may be a main maintainer of the operating system, but a device may include different packages, drivers, and customizations from carriers, manufacturers, and other third parties. Additionally, various open source components used in the operating system are also owned and maintained by various entities. When a vulnerability (defined as a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy) is discovered for any given part of a device, the ecosystem for fixing the vulnerability is so complicated that the process can be extremely slow and, in some cases, the vulnerability is never addressed. Thus, there is a need in the mobile security field to create a new and useful system and method for patching a device through exploitation. This invention provides such a new and useful system and method.

BRIEF DESCRIPTION OF THE FIGURES

[0004] FIG. 1 is schematic representation of system of a preferred embodiment;

[0005] FIG. 2 is a graphical flowchart representation of a method of a preferred embodiment;

[0006] FIG. 3 is a graphical flowchart representation of a method of a variation of a preferred embodiment;

[0007] FIG. 4 is a sequence diagram representing an exemplary implementation of a method of a preferred embodiment;

[0008] FIG. 5 is a schematic representation of an exemplary screenshot of presenting the presence of the identified vulnerability;

[0009] FIG. 6 is a schematic representation of stages of the method including accessing secured vulnerability exploits and securely engaging a reference monitor;

[0010] FIG. 7 is a schematic representation of updating the vulnerability resolution in persistent storage; and

[0011] FIG. 8 is a schematic representation of updating the vulnerability resolution in volatile memory.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0012] The following description of preferred embodiments of the invention is not intended to limit the invention to these preferred embodiments, but rather to enable any person skilled in the art to make and use this invention.

1. System for Patching a Device Through Exploitation

[0013] As shown in FIG. 1, a system for patching a device through exploitation of a preferred embodiment can include a vulnerability tool with a vulnerability assessment component, an exploit engine, and an update component. The system is preferably used to enable capabilities to fix, update, and/or reduce the vulnerabilities of a system. The system can be employed to apply a persistent fix on a computing system but can alternatively be used in maintaining a patched state through a non-persistent computing system patch. The system is preferably used in an operating system that may be characterized as one in which known vulnerabilities are known and known fixes to the vulnerabilities are known, but due to disorganization and lack of incentives, the parties with the privileges to correct such issues do not. Mobile computing systems such as smart phones, tablets, wearable computing devices, open-source hardware systems, vehicular computing systems, and the like are exemplary computing systems where multiple parties are involved. Hardware providers, OS providers, carriers and numerous other parties involved in such multi-party devices are exemplary environments that can employ the use of the system. The components of the system preferably cooperatively identify vulnerabilities and exploit such vulnerabilities to enable capabilities to fix the vulnerabilities. The vulnerability tool can be communicatively coupled to an analysis service and an update data resource. The system can be used in various situations. In one preferred implementation, the system is implemented through a security application that a user of a device selectively downloads and uses. In another preferred implementation, a company can integrate the system as a service on all enterprise managed devices. The system is preferably employed in situations where necessary privileges to update the system are not present or do not facilitate customized and/or accelerated patching of a device

[0014] The vulnerability tool functions as a service or application that can identify and resolve vulnerabilities on a device. The vulnerability tool is preferably an application of a mobile device, which may have a closed and complicated ecosystem that prevents speedy resolution of device vulnerabilities. The vulnerability tool may alternatively be any suitable application or service operable on a device system. The vulnerability tool is preferably downloaded as a third party application on a device. The vulnerability tool may be downloaded in response to a request or purchase by a user/owner of a device. The vulnerability tool may alternatively be automatically deployed by an enterprise or service provider, embedded via an SDK of an existing application, provided in the existing software loadset by an OEM/carrier or other entity, or provided in any suitable manner. The user preferably activates the vulnerability detection capabilities of the application; the user is informed of identified vulnerabilities on a device; and then the user is presented with the option of patching or fixing the vulnerabilities. The vulnerability tool can alternatively or additionally operate in the background. Following installation or initialization of the vulnerability tool, the vulnerability tool can continuously or periodically identify and resolve vulnerabilities on the device. The vulnerability tool then uses a method substantially similar to the one described below to resolve or reduce the vulnerability or vulnerabilities of the device by leveraging the existing vulnerabilities on the device.

[0015] The vulnerability assessment component (VAC) of a preferred embodiment functions to facilitate the identifica-

tion of device vulnerabilities. The VAC may use any suitable approach to identifying a vulnerability. In one preferred embodiment, the VAC cooperates with a remote analysis service to identify vulnerabilities. The VAC is preferably a module, routine, process, or other component of the vulnerability tool that collects software objects to be assessed and transmits vulnerability assessment requests to the analysis cloud service. The VAC additionally can receive the vulnerability assessments and, depending on the application, present the vulnerability assessment result to a user, provide the assessment to an application, automatically initiate patching a vulnerability, or take any suitable action. As many vulnerability assessment techniques may be computationally heavyweight, the VAC preferably off-loads these tasks to the analysis cloud service. The VAC may alternatively implement any suitable additional or alternative techniques to identify a vulnerability such as anti-virus and/or malware detection approaches as would be known by those skilled in the art.

[0016] One preferred embodiment may include an analysis cloud service, which functions to remotely analyze and assess the vulnerability of a plurality of mobile devices that utilize the service. The analysis cloud service is preferably a network-based service. The analysis cloud service is preferably a software component running on a server that is accessible to the mobile device via the network. Such a network may be public or private in nature, and the analysis cloud service may communicate with the VAC using any variety of protocols using any type of network interface deemed appropriate. The analysis cloud service is preferably configured to scale within a cloud-computing environment to meet large volumes of vulnerability requests. The analysis cloud service is tasked with receiving vulnerability assessment requests from a VAC, using the information in the request to identify vulnerabilities using one of its analysis engines, and returning the result back to the VAC over the network or through any suitable means. The analysis cloud service may additionally include a vulnerability storage system to facilitate the detection and analysis of vulnerability in a plurality of devices. A system of the preferred embodiment may alternatively implement the functionality of the analysis locally on a device. A local analysis engine can function substantially similar. Vulnerability updates can be transferred to the local analysis engine to maintain an update record of vulnerabilities. A local analysis engine can perform analysis on a more regular basis due to the lack of network communication required. Additionally, raw information may be more readily inspected locally. A hybrid approach employing a local and cloud analysis engine can additionally be leveraged, depending on different contexts or modes of operation.

[0017] The exploit engine of a preferred embodiment functions to execute an exploitation process. The exploit engine preferably uses at least one of the vulnerabilities identified by the VAC to grant the vulnerability tool heightened permissions within the operating system. The heightened permissions within the operating system preferably include arbitrary privileged code execution. In achieving arbitrary privileged code execution, the capability can be leveraged to install a reference monitor by the update component. The exploit engine may alternatively use any suitable approach to enable the vulnerability tool to gain the capability to apply an update that would otherwise be prohibited by third parties. The exploit engine may access a resource storing a collection of exploitation scripts or routines. An exploitation routine preferably targets at least one vulnerability, more preferably a

vulnerability identified by the analysis engine. In one alternative embodiment, the exploit engine may systematically attempt various exploits and test if the exploit succeeded, which would circumvent the process of analyzing and predicting vulnerabilities.

[0018] The update component of a preferred embodiment functions to apply a patch or system update to resolve or reduce an existing vulnerability. The update component is preferably engaged or initiated upon the vulnerability tool entering an escalated privileges mode and/or a backdoor privilege mode. The escalated privileges mode and/or backdoor privilege mode can be enabled by the exploit engine or through an established reference monitor previously established. The update component will preferably retrieve or access the necessary resources to implement an update within the system. The update component may retrieve the patch data file from an update data resource such as one remotely hosted along with the analysis cloud service or other suitable location. In one preferred embodiment, the update component may include a reference monitor update routine to create or modify a reference monitor. The augmented reference monitor preferably serves as a mechanism to invoke a backdoor privilege mode. The reference monitor may be used as an alternative mechanism to grant the vulnerability tool enhanced capabilities in the operating system. The backdoor privilege mode can be invoked for patching subsequent updates without exploiting a second vulnerability. The reference monitor is preferably a mechanism that enforces an access control policy over subjects' ability to perform operations on objects within the system such that the update component can perform subsequent updates. The reference monitor is preferably secured so that only the update component and/or other suitable components of the system can use the reference monitor to acquire desired privileges to execute an update.

2. Method for Patching a Device Through Exploitation

[0019] As shown in FIG. 2, a method for patching a device through exploitation of a preferred embodiment can include identifying a vulnerability in a system **S110**; entering an operating mode of escalated privileges through the identified vulnerability **S120**; and while in the escalated privileges mode, updating the system with a vulnerability resolution **S130**. The method functions to enable outside entities to update or patch a closed system to resolve a vulnerability. The closed system is preferably a computing device (e.g., a mobile phone, a tablet, a wearable computing device, a home automation computing system, a smart device, a gaming system, a home entertainment system, and the like) in a closed system administration ecosystem, where patching a system is managed by a limited set of parties. In some situations, a device may have an existing vulnerability of which a solution is known (i.e., a patch exists), but due to logistical issues, the patch or solution is not obtainable from the device owner. The logistical issue may arise from a party with the necessary privileges (e.g., a carrier) not acting. Thus, an implementer of the method of a preferred embodiment can circumvent patching restrictions by exploiting a vulnerability in order to resolve or mitigate the vulnerability. The method is preferably implemented by vulnerability tool of mobile computing device or other suitable device as described above and may additionally use and access analysis, exploitation instructions and patching resources from within the vulnerability tool or hosted remotely.

[0020] Step S110, which includes identifying a vulnerability in a system, functions to find at least one vulnerability or weakness on the device. Preferably, vulnerabilities are identified for an entire system, but the vulnerabilities may alternatively be a set of vulnerabilities for a particular component or set of components of a device such as for a service or an application within an operating system of a computing device. Preferably the checked system is a mobile operating system. Alternatively, the checked system may be a particular application (e.g., a browser, communication framework, or any suitable component). Identifying vulnerability preferably includes using and processing a vulnerability assessment request in a remote analysis cloud service as is described in U.S. patent application Ser. No. 13/601,409, filed 31 Aug. 2012, which is hereby incorporated in its entirety by this reference. The method preferably includes collecting data object identifiers of the inspected system. Data objects identifiers can include version numbers and type identifiers of system components (e.g., software versions, hardware model identifiers, file names, application names, and the like). The vulnerability analysis may alternatively be performed and implemented on the device without accessing an analysis cloud service. Preferably, identifying a vulnerability includes disassembling executable code into native machine code and detecting unpatched vulnerabilities. For example, a file such as the volume manager file may be disassembled into its native machine code and analyzed to determine whether a known vulnerability, such as the gingerbreak vulnerability for Android devices, has been patched. Code analysis routines can preferably be developed and deployed to an analysis cloud service at any suitable time, which enables the analysis cloud service to provide current vulnerability assessment to all devices at the time of their vulnerability assessment. Alternatively or additionally, identifying a vulnerability may include querying a map of object identifiers to vulnerability assessments. Querying a map of object identifiers to vulnerability assessments can include collecting data object identifiers and searching for vulnerabilities mapped to the object identifiers. The object identifiers can be a version number, a product/application name, a file name, executable code excerpts, or any suitable object identifier. The object identifier can additionally be a combination of object identifiers such as a first software version number, a file name, and a device model number may be used as a single object identifier for a vulnerability that targets that particular device configuration. For example, the version number of software may have been included as the object identifier, and that object identifier may be used to look up a known vulnerability assessment. More preferably, a cryptographic hash of a code segment may be used to query a storage system for an assessment previously calculated for the code segment. The analysis cloud service preferably includes a combination of approaches to identify a vulnerability. Identifying a vulnerability may additionally or alternatively include various anti-virus and malware approaches as would be known by those skilled in the art.

[0021] The vulnerability can assessment can be completed locally at the device, but the vulnerability assessment can alternatively be completed remotely such as within a vulnerability assessment cloud service as shown in FIG. 4. Identifying a vulnerability in a system can include transmitting a data object identifier to a remote vulnerability assessment service and receiving a vulnerability assessment that identifies at least one vulnerability of the computing device. The vulnerability assessment cloud service can use information

from multiple users and can quickly distribute new vulnerability assessment information. Additionally, vulnerability assessments can retroactively be applied. For example, a device may submit an object identifier. At the time of submission, a vulnerability may not exist or be known, and no vulnerability is identified. The object identifier is preferably stored as an object identifier record and associated with the particular device. At a later time, a vulnerability may be discovered that maps to that object identifier. Action can then be taken based on stored object identifier, which functions to simplify the collection of object identifiers for a device. The method can include identifying a vulnerability in a system from an object identifier record, and at the device asynchronously receiving at least one vulnerability assessment that identifies at latent vulnerability. Here asynchronous describes the submission or collection of the information characterizing the vulnerability (e.g., the data object identifier or more specifically the executable code sample) is collected at a time substantially earlier time (e.g., days, weeks, or weeks prior to identifying the vulnerability). Latent similarly describes the nature of the vulnerability being present in the system (which may have been previously scanned) but not identified. Object identifier records can be processed and queried whenever a new vulnerability to object identifier is added to the system. If a vulnerability is detected for an object identifier, the associated device or devices can be messaged, notified, triggered, or otherwise prompted to initiate vulnerability patching.

[0022] The identified vulnerabilities are preferably those that can be exploited for escalated privileges so that the system may be patched in Steps S120 and S130. In one example implementation, the method may be applied to a vulnerability tool for Android operating system wherein an analysis engine may include algorithms and heuristics to identify privilege escalation vulnerabilities due to allowing a device to be rooted (e.g., ASHMEM vulnerability), neglecting to check if messages are from a trusted source (e.g., Exploit or Gingerbreak vulnerability), privilege levitating (e.g., Levitator vulnerability), writing to arbitrary memory address space (e.g., Mempoandroid vulnerability), overwhelming a system with command arguments (e.g., Android ZergRush vulnerability), neglecting to check return values enabling root shells (e.g., Zimperlich vulnerability), and/or any suitable vulnerability or exploits. As vulnerabilities are constantly changing as new ones are found, and new ones introduced. The set of identifiable vulnerabilities is preferably periodically updated.

[0023] The method can additionally include presenting the presence of the identified vulnerability S112, which functions to inform, alert, record, or track the vulnerabilities that have been detected and that are, will be, or can be addressed. The alerting of the presence of the identified vulnerability can be applied to informing a user, notifying an administrator, notifying involved parties, notifying other installed applications or services. In one variation, the identified vulnerabilities can be detected by a user-run application. The application can display a vulnerability summary view as shown in FIG. 5 or play a suitable vulnerability detection alert. In a related variation, the application can use a notification or messaging service of the device to alert the user such as if the application is a background application that is not commonly opened by the user. The vulnerability summary can include a count of summaries; an identifier of the vulnerability (e.g., a name or internal identifier); the services, applications, functionality and/or other aspects that are vulnerable; any known resolu-

tions or options to patch the vulnerability through software updates; and/or any other suitable information. The vulnerability summary can additionally include selectable user interface options (e.g., buttons or selectable actions) that can be used to receive a request to apply a vulnerability resolution. The user interface options can additionally include an option to install a reference monitor. The user interface options can additionally include an option to provide device vulnerability data to an outside entity such as the vulnerability assessment cloud service. The vulnerability data can include anonymized vulnerability status of the device. The vulnerability data can be used in further improving vulnerability detection and assessment. The vulnerability data can additionally or alternatively be used in informing and prompting involved parties to take action. As mentioned above, other involved services or applications on the device may be programmatically notified. For example, some applications (e.g., a financial transaction based app) may register for the security status of the device such that privileges can be changed depending on the present vulnerabilities and/or the resolution/patching of vulnerabilities.

[0024] Step S120, which includes entering an operating mode of escalated privileges through the identified vulnerability, functions to benevolently leverage the vulnerability to enable necessary capabilities to fix or lessen the threat. Entering an operating mode of escalated privileges preferably includes selecting an appropriate vulnerability exploitation and executing the vulnerability exploitation. If a plurality of vulnerabilities were identified, the selected vulnerability for exploitation may be made based on the robustness, simplicity, enabled capabilities after the exploitation, or other properties of the exploit. For example, one vulnerability may only enable patching one particular vulnerability, but a second vulnerability may enable patching of multiple vulnerabilities and installing a reference monitor. In another variation, a plurality of vulnerabilities may be selected for combined exploitation. The process of exploitations may be characterized in exploit scripts of modules. The exploit modules may be cryptographically or suitably secured such that an exploit module is substantially limited to execution by the vulnerability tool—only authorized modules can preferably execute the exploit on the device. For example, only digitally signed requests by an authorized party may be limited to access the exploit module as shown in FIG. 6. Third parties or attackers could be prevented from abusing the system and tricking a device into executing malicious code. The process of the exploit module is preferably dependent for a given vulnerability. In addition to signing for integrity of the exploit modules, the exploit modules may additionally be encrypted to achieve confidentiality. The exploit modules can be correspondingly decrypted on the device (e.g., by a reference monitor).

[0025] The exploit preferably grants the vulnerability tool the necessary privileges to patch the vulnerability. The privileges are preferably escalated privileges such as those for a superuser, root account, system administrator, kernel level account, higher privileged services/drivers, privileges of another user/application/service, and/or other suitable modes of increased permissions within the device. Often these are system privileges that are reserved for carriers or manufacturers. The privileges may be based on the identified vulnerabilities. Depending on the vulnerability, different approaches may be taken. The privilege escalation is preferably a variety of vertical privilege escalation to obtain

increased privileges. However, the privilege escalation may alternatively be horizontal to receive privileges of another application or other suitable component with the same level but different areas of privileges. Some exemplary exploits can include leveraging a buffer overflow vulnerability to perform arbitrary privileged code execution. In another example, the exploit uses application-level privilege escalation, which can include confused deputy exploits and collusion exploits. A confused deputy exploit leverages vulnerable interfaces of another privileged (but confused) application. Benign applications can be vulnerable to exploitation by exposing vulnerable interfaces. In some cases over-privileged benign applications can be used to obtain increased privileges. Collusion attacks can leverage and temporarily cooperate with malicious applications to combine permissions to allow, grant, or enable performing actions beyond respective individual privileges. Colluding application can communicate directly or exploit covert or overt channels in core system components.

[0026] Step S130, which includes updating the system with a vulnerability resolution while in the escalated privileges mode, functions to use the privileges granted through the exploit to resolve or reduce vulnerabilities. Updating the system preferably includes downloading, accessing, and/or selecting a patch or software update that can be applied to the system. When an update is obtained, the patch can preferably be applied. The updating can target only the identified vulnerability; alternatively, updating may additionally install or update other system changes. Preferably at least the vulnerability that is exploited is patched. More than two patches may additionally be applied. If a plurality of vulnerabilities were identified in Step S110, the system can be updated with a patch for the plurality of vulnerabilities. In one implementation, the vulnerability tool (or a reference monitor or other related component) can patch the vulnerable software component in a persistent manner. For example, the vulnerability tool can write to a software library that is on a persistent disk/storage or non-volatile memory of the device as shown in FIG. 7. In other words, the vulnerability resolution is persistently applied in persistent storage of the device. In an alternative implementation, the vulnerability tool or other component can perform a soft patch, wherein the vulnerability tool can write a patch to system memory. Patching system memory (i.e., a soft patch) is preferably applied in non-persistent, volatile memory. The resolution is removed when the device enters a powered down state or the memory is overwritten (e.g., when a soft patched application is deallocated). Performing a soft patch may afford advantages in stability. For example, if a patch was applied improperly, restarting or reloading the application or device may recover the unpatched state. Additionally, the soft patch can be configured to preemptively patch or update vulnerabilities. For example a reference monitor can be configured to execute upon device boot to patch vulnerabilities in memory before potentially malicious code can run. A reference monitor can be configured to be initiated and engaged upon restarting or turning on the device, service, application, or other vulnerable component. The reference monitor can then apply a soft patch in substantially similar as previously performed as shown in FIG. 8. Alternatively a script or patching routine applies the patch upon restarting the device, service, or application, or other vulnerable component. Re-patching can additionally be controllable through application configuration. The reference monitor (or other suitable soft-patch initializa-

tion tool) can be selectively configured to re-patch one or all soft-patched vulnerabilities on the device on reboot, enabling the device to return to an original configuration. Additionally the application configuration can be applied to enable hard patch (i.e., patch in persistent memory) after the soft-patch has been verified to not conflict with normal operation.

[0027] Additionally, the method of a preferred embodiment may include establishing a reference monitor **S140**; enabling a backdoor privilege mode through the reference monitor **S150**; and updating the system with a resolution for at least a second exploitation while in the backdoor privilege mode **S160** as shown in FIG. 3. The use of a reference monitor preferably functions to establish an access channel to apply subsequent updates to the system. The use of the reference monitor backdoor further functions to enable the vulnerability tool to patch or substantially reduce the vulnerability that was exploited to enable the patching, while ensuring that the vulnerability tool maintains a way to supply subsequent updates. When a reference monitor is established, the vulnerability tool can apply updates even for a vulnerability without a suitable exploit (e.g., cannot be used to heighten privileges). The vulnerability can enable the backdoor privilege mode using the reference monitor in place of executing an exploit.

[0028] Step **S140**, which includes establishing a reference monitor **S140**, functions to update the system with backdoor access to administer system updates. The reference monitor preferably impacts the reference validation mechanism of the system, which enforces an access control policy over subjects ability to perform read, write and other operations within the system. A reference validation mechanism of the reference monitor is preferably always invoked, which can avoid bypassing the security policy. The reference validation is preferably tamper proof. The reference monitor is preferably configured to allow the vulnerability tool to have the necessary operational permissions to subsequently update the system. In one variation, this would include granting the vulnerability tool system permissions similar to those that would be invoked by a carrier or other controlling entity. Alternatively, the reference monitor may be updated with a latent backdoor mechanism that can be securely activated by the vulnerability tool. Preferably, the reference monitor is not exposed to vulnerabilities by outside entities and is only accessible by the vulnerability tool. For example, a reference monitor may be configured to only accept instructions that are digitally signed by an authorized party, which functions to ensure that unauthorized parties are not able to invoke it in a malicious way. In an alternative embodiment, the vulnerability may establish an alternative form of backdoor in addition to or as an alternative to a reference monitor. For example a static key or password may be used to grant access to the reference monitor or any suitable proof of authentication.

[0029] Step **S150**, which includes enabling a backdoor privilege mode through the reference monitor, functions to use the heightened/escalated privileges afforded by the established reference monitor. The backdoor privileges are preferably heightened privileges. As mentioned above, the backdoor privilege mode may be inherently enabled through the establishment of the reference monitor. For example, augmenting a reference monitor may grant the vulnerability tool a constant state of heightened privileges. Alternatively, the vulnerability tool may have to enable an established mechanism to grant heightened privileges. The heightened privileges can be sustained for at least the time duration used to complete updating the system with a resolution for at least a one exploit-

ation. The enabling of backdoor privileges preferably escalates privileges such as changing privileges of the vulnerability tool to those for a superuser, root account, system administrator, kernel level account, higher privileged services/drivers, privileges of another user/application/service, and/or other suitable modes of increased permissions within the device. The reference monitor is preferably secured so that only the vulnerability tool or other suitable system component can access and enable the heightened privilege mode. The backdoor privilege mode is preferably used when detecting a subsequent vulnerability. For example, a first vulnerability is detected at time **t1**; the first vulnerability is patched and a reference monitor is set up; a second vulnerability is detected at time **t2**, where **t2** comes after **t1**; the established reference monitor is used to invoke heightened privileges without executing an exploit; and the second vulnerability is patched. The detection of the second vulnerability can happen for a vulnerability of a set of vulnerabilities detected with the first vulnerability, during a second local scan of new data objects of the device, scanning for latent vulnerability in data object records using new vulnerability information, or in response to any patching prompt.

[0030] Step **S160**, which includes updating the system with a resolution for at least a second exploitation while in the backdoor privilege mode, functions to use the privileges granted through the exploit to resolve or reduce vulnerabilities. Step **S160** is preferably substantially similar to Step **S130**. However, the resolution for at least a second exploitation can be made without entering an operating mode of escalated privileges through the second exploitation (though that variation may alternatively be used if the required privileges cannot be obtained other than through the second exploitation). Instead, escalated privileges are preferably granted through enabling escalated privileges through the reference monitor. Alternatively, escalated privileges can be granted through a backdoor of a previously resolved vulnerability. In any case one of the above variations is used to escalate privileges such that a vulnerability resolution can be installed, patched, or otherwise enabled. The method can additionally include exiting the operating mode of escalated privileges. After the vulnerability is completed, the reference monitor backdoor may be eliminated or updated to secure access for a subsequent use. For example, the heightened privileges of the backdoor privilege mode are removed and returned to normal operating privileges. The backdoor privileges can be reactivated at a later time when an update or other suitable action is necessitated. Additionally, the reference monitor can itself be removed such that that mechanism can no longer be used. If a reference monitor is removed, subsequent identified vulnerabilities can be resolved through executing a corresponding exploit of the vulnerability and/or establishing a second reference monitor as described above.

[0031] The system and methods of the preferred embodiment and variations thereof can be embodied and/or implemented at least in part as a machine configured to receive a computer-readable medium storing computer-readable instructions. The instructions are preferably executed by computer-executable components preferably integrated with the vulnerability tool. The computer-readable medium can be stored on any suitable computer-readable media such as RAMs, ROMs, flash memory, EEPROMs, optical devices (CD or DVD), hard drives, floppy drives, or any suitable device. The computer-executable component is preferably a general or application specific processor, but any suitable

dedicated hardware or hardware/firmware combination device can alternatively or additionally execute the instructions.

[0032] As a person skilled in the art will recognize from the previous detailed description and from the figures and claims, modifications and changes can be made to the preferred embodiments of the invention without departing from the scope of this invention defined in the following claims.

What is claimed is:

1. A method comprising:
 - identifying a vulnerability in a computing device;
 - accessing a vulnerability exploitation mapped to the identified vulnerability;
 - at the computing device, executing the vulnerability exploitation and entering an operating mode of escalated privileges; and
 - while in the operating mode of escalated privileges, updating the system with a vulnerability resolution.
2. The method of claim 1, further comprising collecting data object identifiers of the computing device; and wherein identifying a vulnerability in a computing device comprises querying a mapping of object identifiers to vulnerabilities.
3. The method of claim 1, further comprising presenting the presence of the identified vulnerability.
4. The method of claim 1, wherein accessing the vulnerability exploit comprises digitally signing a request to access a cryptographically secured vulnerability exploit.
5. The method of claim 1, wherein updating the system with a vulnerability resolution comprises persistently applying the vulnerability resolution on the device.
6. The method of claim 1, wherein updating the system with a vulnerability resolution comprises applying the vulnerability resolution on the device in non-persistent memory; and further comprising reactivating the vulnerability resolution on the device in the non-persistent memory during restart of the device.
7. The method of claim 1, further comprising while in the operating mode of escalated privileges, establishing a reference monitor; and further comprising identifying a second vulnerability in a computing device; entering an operating mode of backdoor escalated privileges through the reference monitor; and while in the operating mode of backdoor escalated privileges, updating the system with a vulnerability resolution of the second vulnerability.
8. The method of claim 1, wherein more than one vulnerability is identified; and
 - wherein updating the system with a vulnerability resolution comprises updating the system with a vulnerability resolution for more than one vulnerability.
9. The method of claim 1, wherein identifying a vulnerability in a computing device further comprises collecting data object identifiers of the computing device, transmitting the data object identifiers to a remote vulnerability assessment cloud service; and receiving a vulnerability assessment that identifies at least one vulnerability of the computing device.
10. The method of claim 9, wherein receiving a vulnerability assessment that identifies at least one latent vulnerability of the computing device occurs asynchronously to the transmitting the data object identifiers.
11. A method comprising:
 - identifying a first vulnerability in a system;
 - exploiting the first vulnerability and granting escalated code execution privileges;

- establishing a reference monitor during the escalated code execution privileges;
- identifying a second vulnerability in the system;
- entering an operating mode of escalated privileges through the reference monitor; and
- while in the operating mode of escalated privileges, updating the system with a vulnerability resolution of the second vulnerability.

12. The method of claim 11, wherein exploiting the first vulnerability comprises digitally signing a request to access a cryptographically secured vulnerability exploit of the first vulnerability and receiving the granting escalated code execution privileges through executing the cryptographically secured vulnerability exploit.

13. The method of claim 11, wherein entering an operating mode of escalated privileges through the reference monitor comprises authenticating a vulnerability tool wherein the vulnerability tool completes updating the system with a vulnerability resolution of the second vulnerability.

14. The method of claim 11, further comprising presenting the presence of the identified vulnerability through a user interface of the system.

15. The method of claim 11, wherein updating the system with a vulnerability resolution comprises persistently applying the vulnerability resolution on the device.

16. The method of claim 11, wherein updating the system with a vulnerability resolution comprises applying the vulnerability resolution on the system in non-persistent memory; and further comprising upon refreshing the system, entering an operating mode of escalated privileges through the reference monitor and reactivating the vulnerability resolution on the device in the non-persistent memory.

17. The method of claim 11, wherein the vulnerability is a system level vulnerability.

18. The method of claim 11, wherein the vulnerability is an application level vulnerability.

19. A method for patching a computing device in a closed system administration ecosystem comprising:

- collecting data object identifiers of the computing device at a first instance;
- querying a map of object identifiers to vulnerabilities and identifying at least one vulnerability;
- accessing a vulnerability exploitation mapped to the identified vulnerability;
- at the computing device, executing the vulnerability exploitation and entering an operating mode of escalated privileges; and
- while in the operating mode of escalated privileges, updating the system with a vulnerability resolution of the at least one vulnerability and establishing a reference monitor;
- collecting data object identifiers of the computing device at a second instance;
- identifying a second vulnerability from the data object identifiers of the computing device at a second instance;
- entering an operating mode of backdoor escalated privileges through the reference monitor; and
- while in the operating mode of backdoor escalated privileges, updating the system with a vulnerability resolution of the second vulnerability.

* * * * *