US 20250260721A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2025/0260721 A1

Vissamsetty et al. (43) Pub. Date: Aug. 14, 2025

(54) **DECEIVING ATTACKERS ACCESSING ACTIVE DIRECTORY DATA**

(71) Applicant: **SentinelOne, Inc.**, Mountain View, CA (US)

(72) Inventors: **Venu Vissamsetty**, San Jose, CA (US); **Anil Gupta**, Bangalore (IN); **Harinath Vishwanath Ramchetty**, Bangalore (IN)

(73) Assignee: **SentinelOne, Inc.**, Mountain View, CA (US)

(21) Appl. No.: **19/056,083**

(22) Filed: **Feb. 18, 2025**

Related U.S. Application Data

(63) Continuation of application No. 18/173,611, filed on Feb. 23, 2023, now Pat. No. 12,261,884, which is a continuation of application No. 16/543,189, filed on Aug. 16, 2019, now Pat. No. 11,616,812, which is a continuation-in-part of application No. 15/383,522, filed on Dec. 19, 2016, now Pat. No. 10,599,842.

Publication Classification

(51) **Int. Cl.**

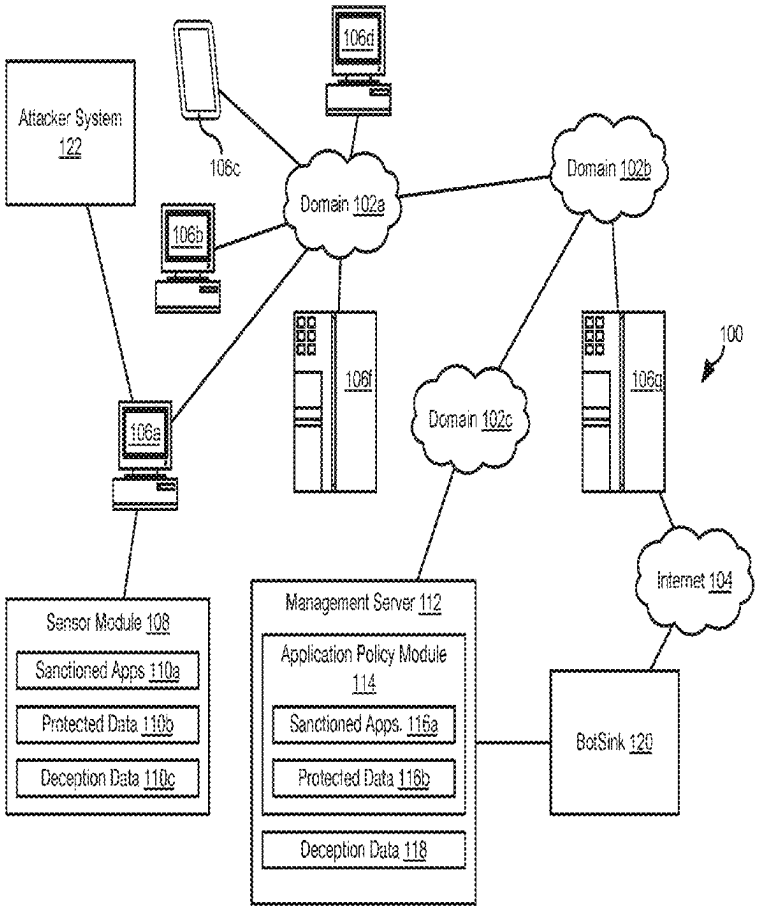
H04L 9/40	(2022.01)
G06F 21/55	(2013.01)
G06F 21/56	(2013.01)

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

CPC ***H04L 63/1491*** (2013.01); ***G06F 21/55*** (2013.01); ***G06F 21/566*** (2013.01); ***H04L 63/14*** (2013.01); ***H04L 63/1416*** (2013.01); ***H04L 63/1441*** (2013.01); *H04L 63/10* (2013.01)

(57) **ABSTRACT**

Endpoints in a network execute a sensor module that intercepts commands. The sensor module compares a source of commands to a sanctioned list of applications received from a management server. If the source does not match a sanctioned application and the command is a write or delete command, the command is ignored and a simulated acknowledgment is sent. If the command is a read command, deception data is returned instead. In some embodiments, certain data is protected such that commands will be ignored or modified to refer to deception data where the source is not a sanctioned application. The source may be verified to be a sanctioned application by evaluating a certificate, hash, or path of the source. Responses from an active directory server may be intercepted and modified to reference a decoy server when not addressed to a sanctioned application.



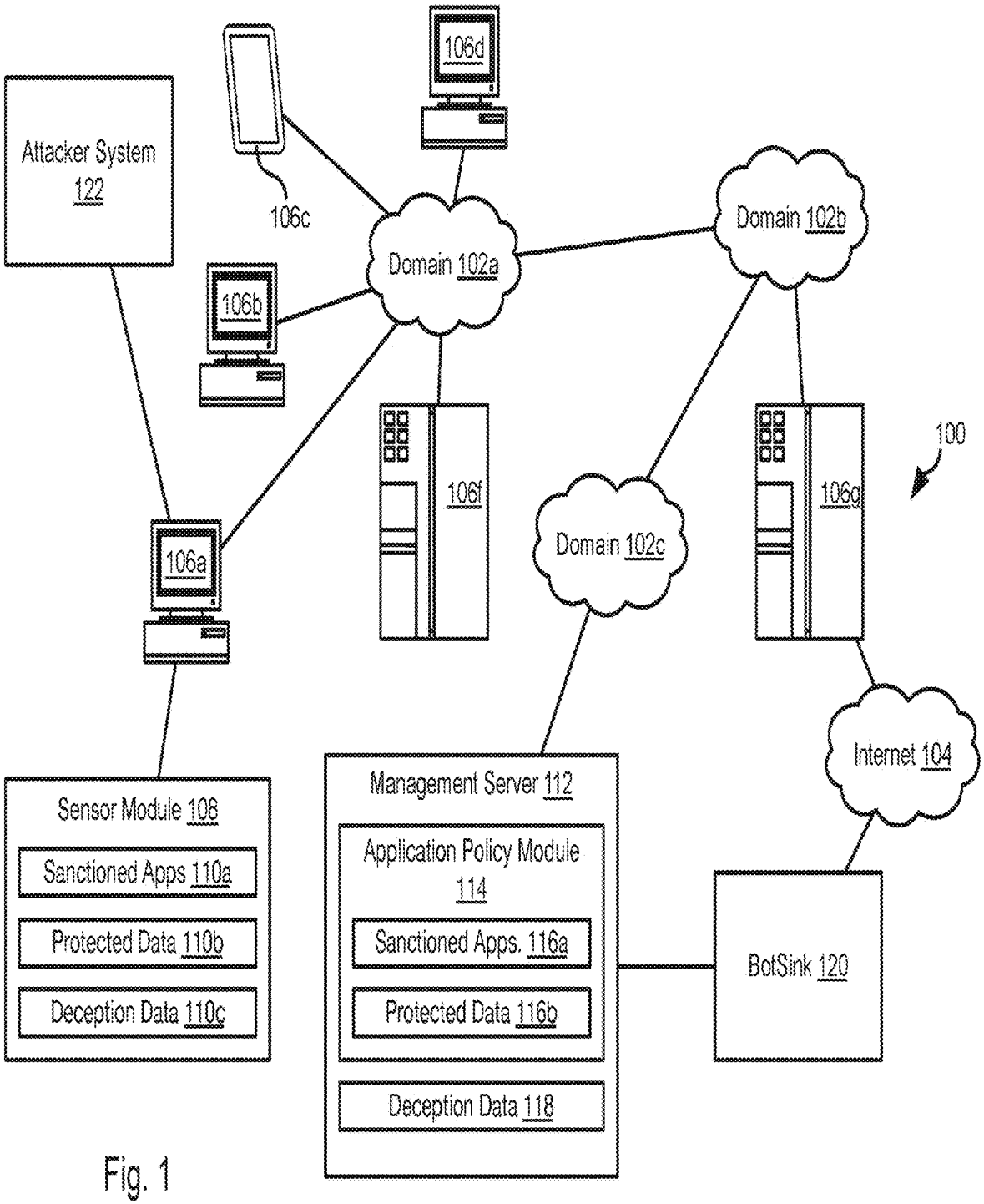

Attacker System 122
106c
106d
106b
Domain 102a
Domain 102b
106f
106a
Domain 102c
106g
100
Internet 104
Sensor Module 108
Sanctioned Apps 110a
Protected Data 110b
Deception Data 110c
Management Server 112
Application Policy Module 114
Sanctioned Apps. 116a
Protected Data 116b
Deception Data 118
BotSink 120

Fig. 1

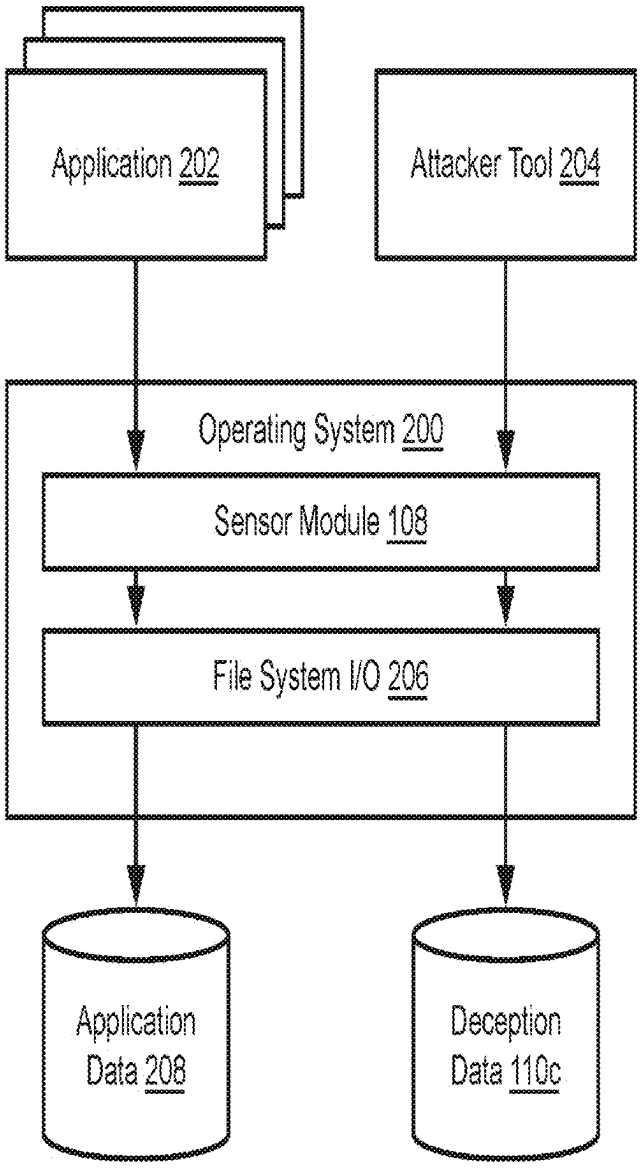

Application 202
Attacker Tool 204
Operating System 200
Sensor Module 108
File System I/O 206
Application Data 208
Deception Data 110c

Fig. 2

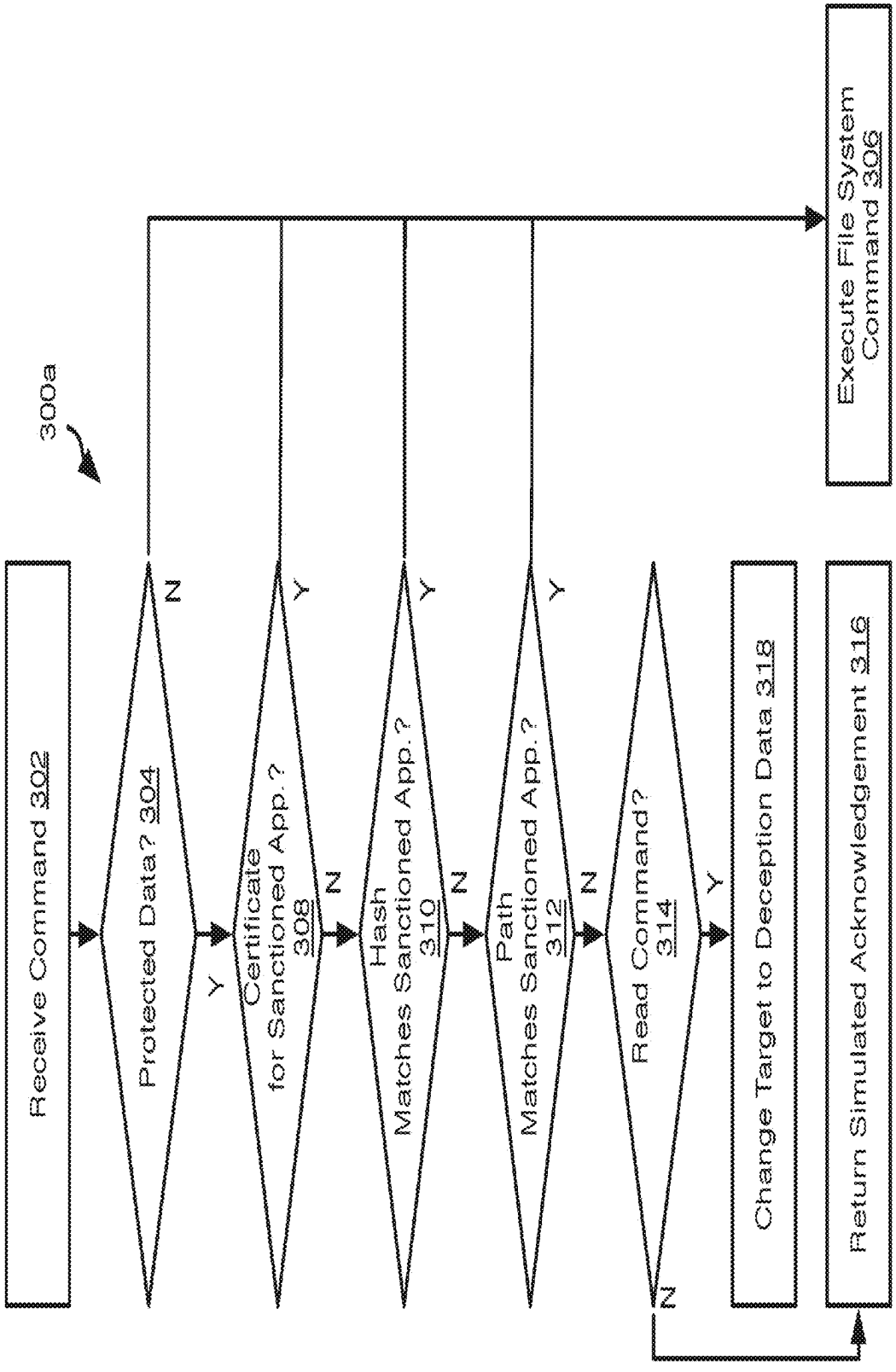

300a
Receive Command 302
Protected Data? 304
Y
N
Certificate for Sanctioned App.? 308
Y
N
Hash Matches Sanctioned App.? 310
Y
N
Path Matches Sanctioned App.? 312
Y
N
Read Command? 314
Y
N
Change Target to Deception Data 318
Return Simulated Acknowledgement 316
Execute File System Command 306

Fig. 3A

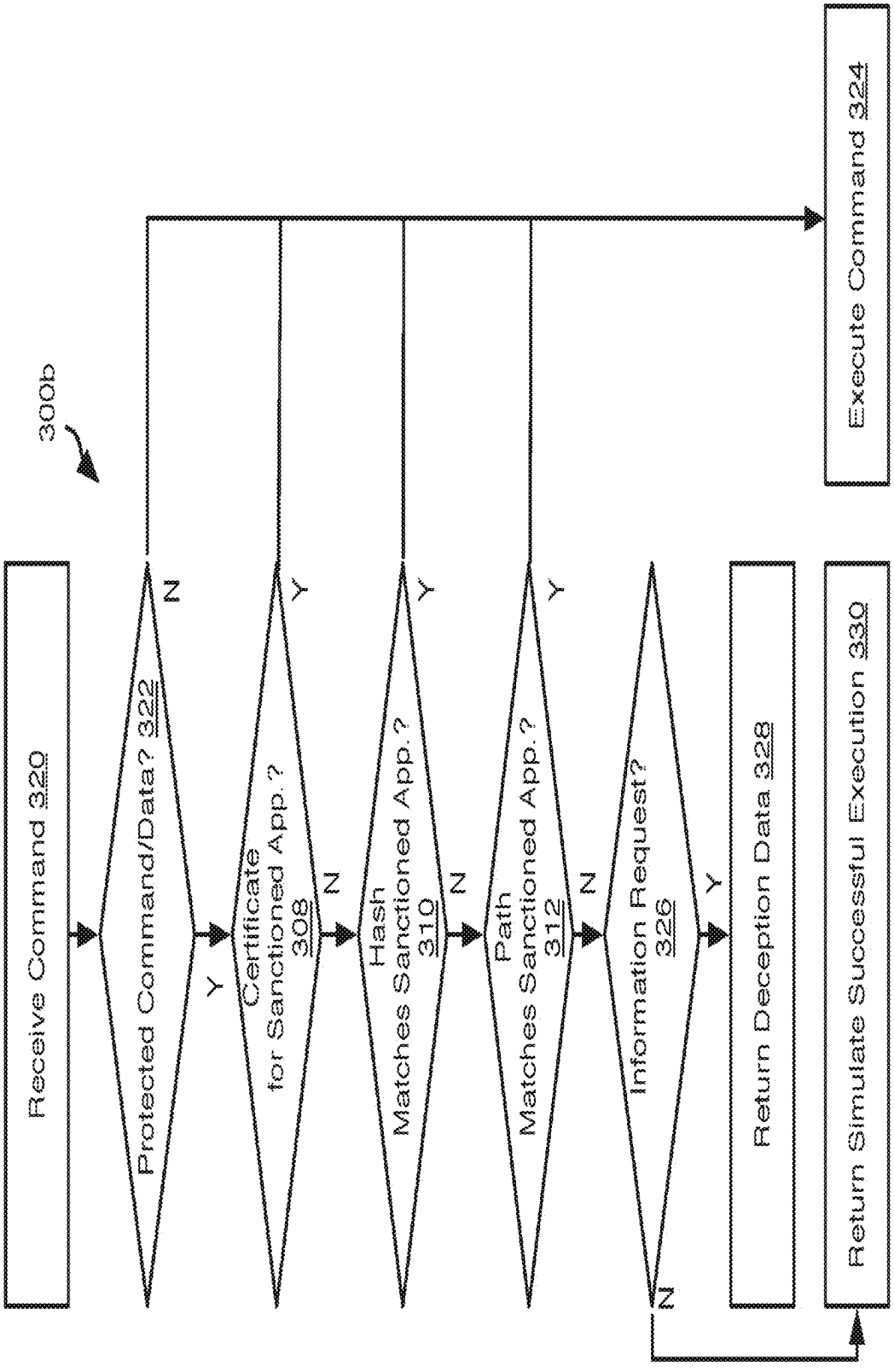

300b
Receive Command 320
Protected Command/Data? 322
Y
N
Certificate for Sanctioned App.? 308
Y
N
Hash Matches Sanctioned App.? 310
Y
N
Path Matches Sanctioned App.? 312
Y
N
Information Request? 326
Y
N
Return Deception Data 328
Return Simulate Successful Execution 330
Execute Command 324

Fig. 3B

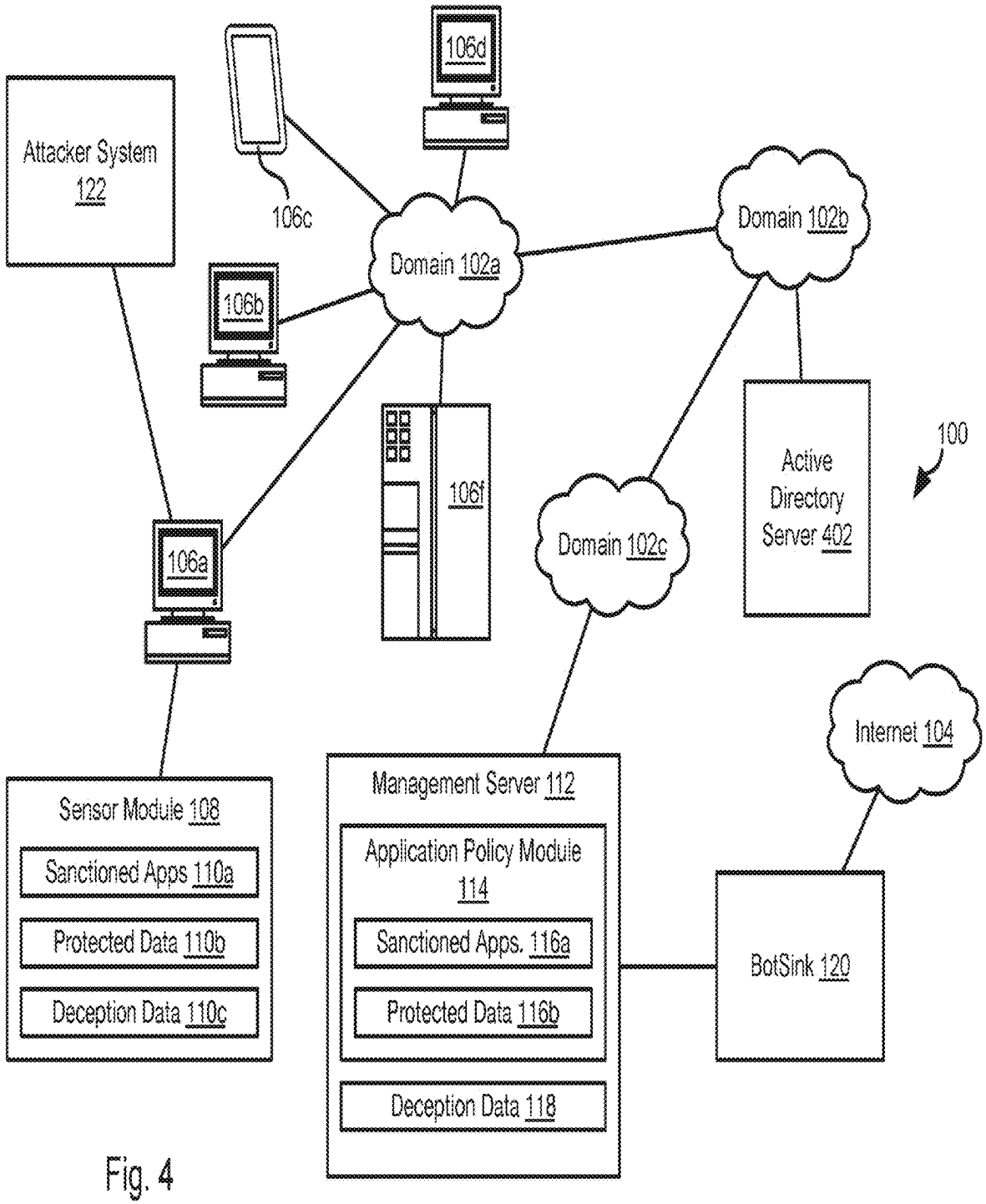

Attacker System 122
106c
106d
106b
Domain 102a
Domain 102b
106f
106a
Domain 102c
Active Directory Server 402
100
Internet 104
Sensor Module 108
Sanctioned Apps 110a
Protected Data 110b
Deception Data 110c
Management Server 112
Application Policy Module 114
Sanctioned Apps. 116a
Protected Data 116b
Deception Data 118
BotSink 120

Fig. 4

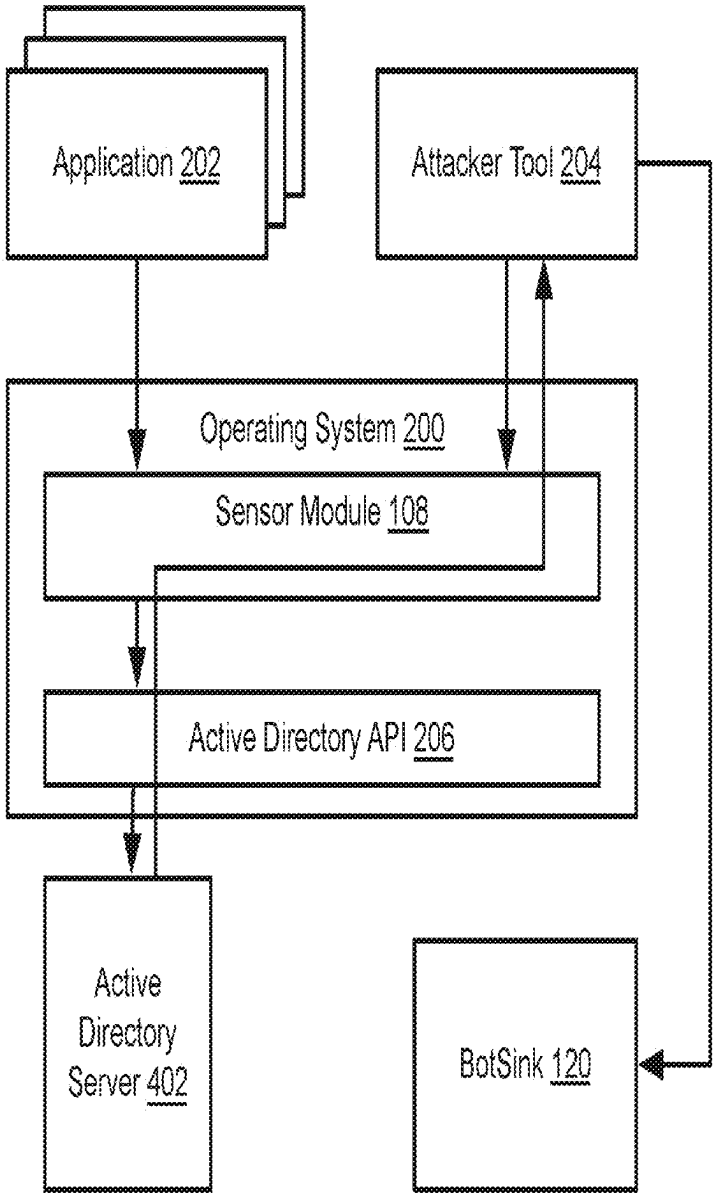

Application 202
Attacker Tool 204
Operating System 200
Sensor Module 108
Active Directory API 206
Active Directory Server 402
BotSink 120

Fig. 5

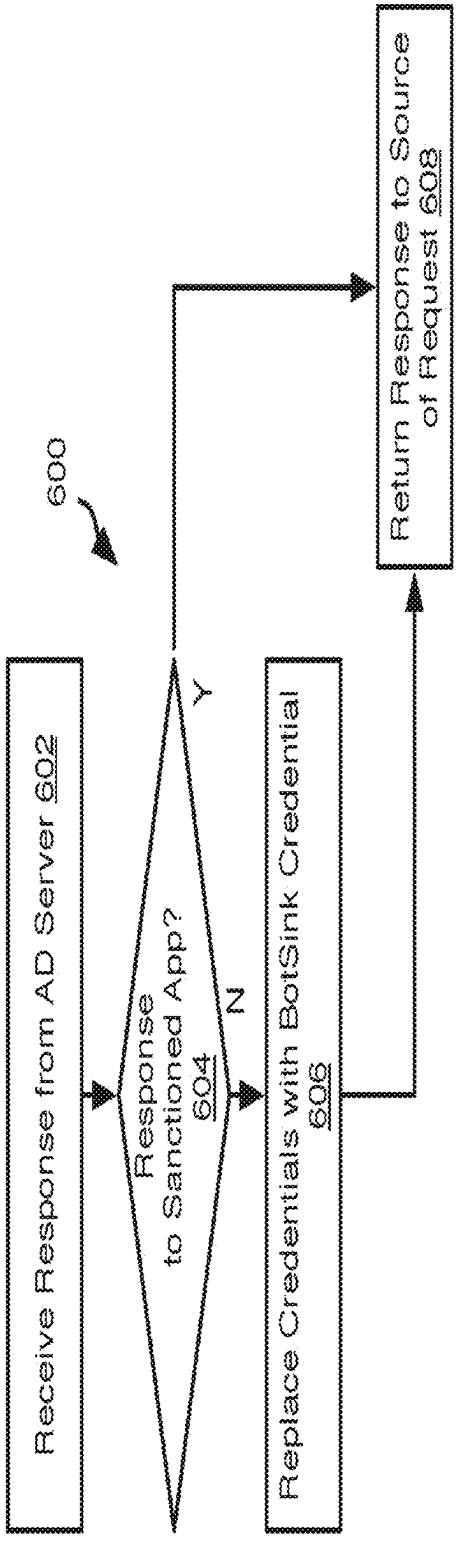

600
Receive Response from AD Server 602
Response to Sanctioned App? 604
Y
N
Replace Credentials with BotSink Credential 606
Return Response to Source of Request 608

Fig. 6

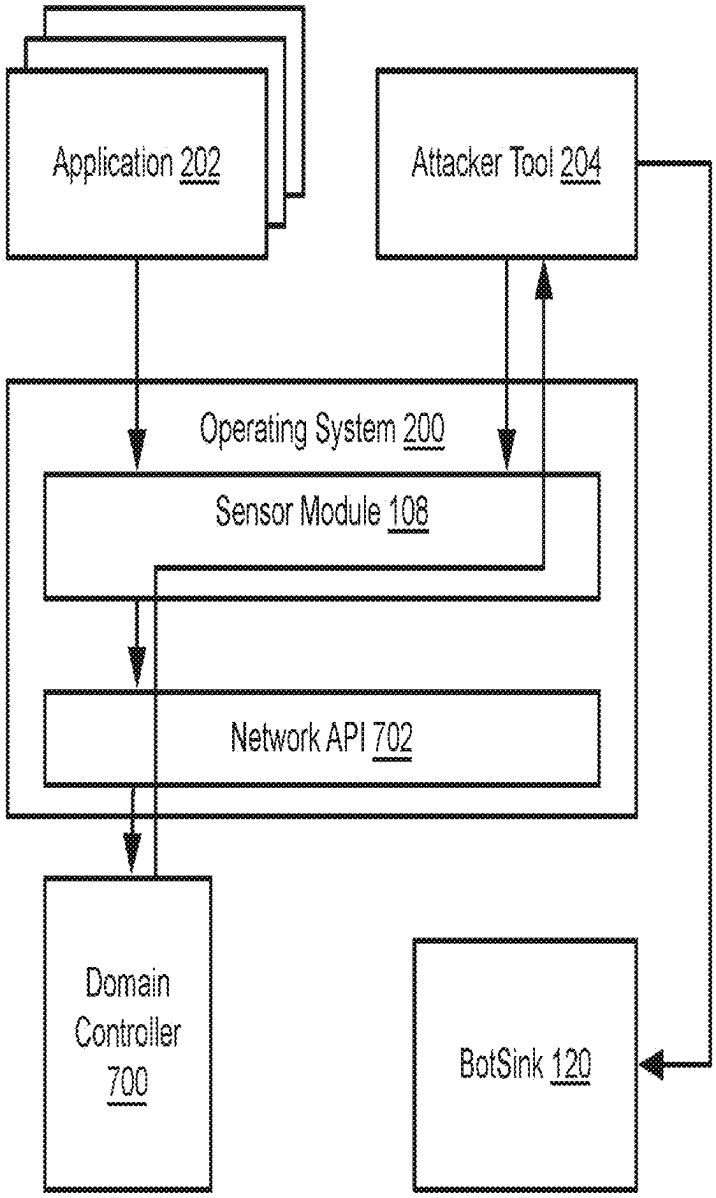

Application 202
Attacker Tool 204
Operating System 200
Sensor Module 108
Network API 702
Domain Controller 700
BotSink 120

Fig. 7

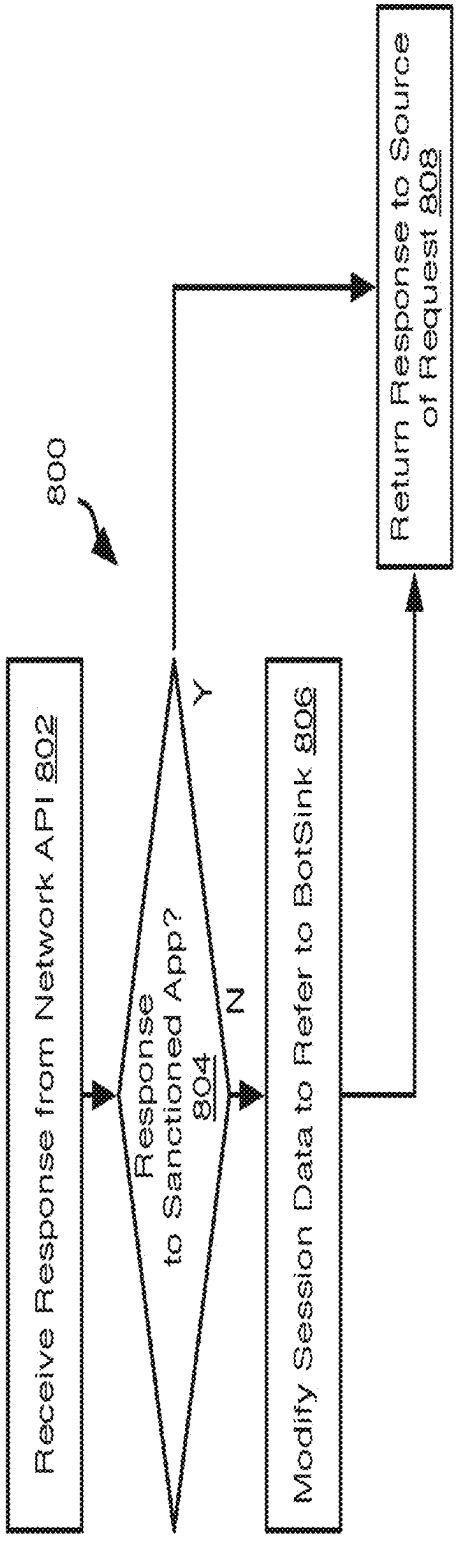

800
Receive Response from Network API 802
Response to Sanctioned App? 804
Y
N
Modify Session Data to Refer to BotSink 806
Return Response to Source of Request 808

Fig. 8

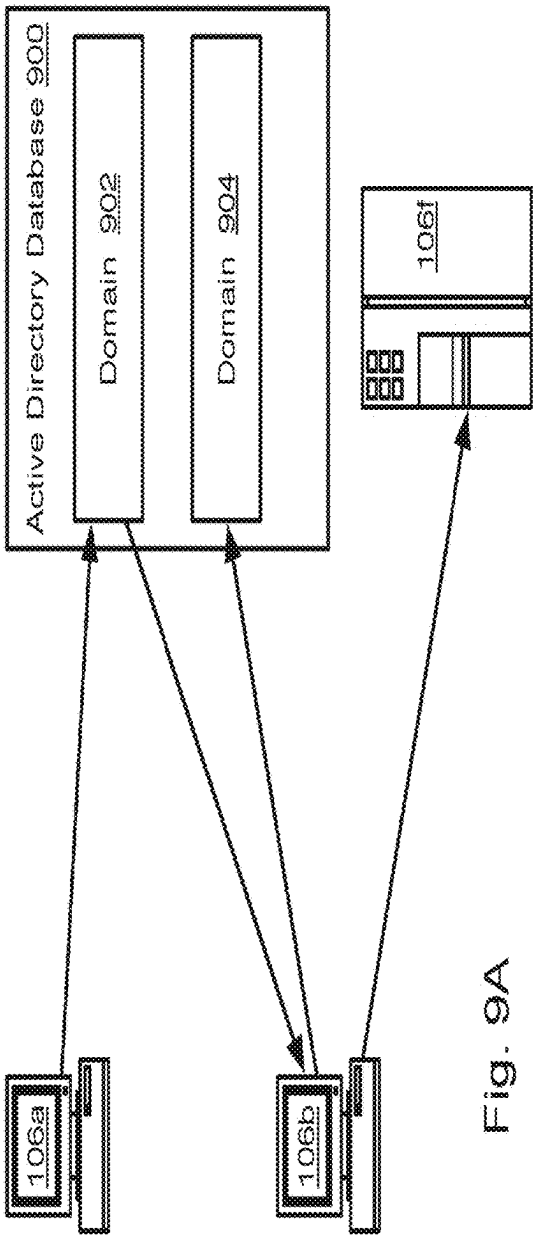

Active Directory Database 900
Domain 902
Domain 904
106f
106a
106b

Fig. 9A

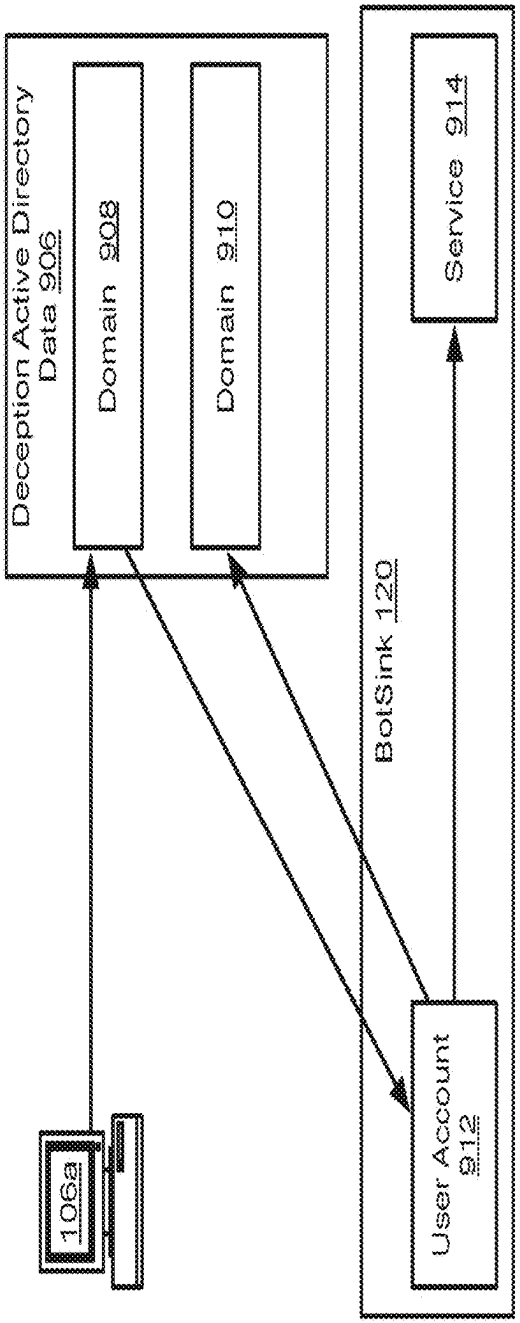

Deception Active Directory Data 906
Domain 908
Domain 910
BotSink 120
Service 914
User Account 912
106a

Fig. 9B

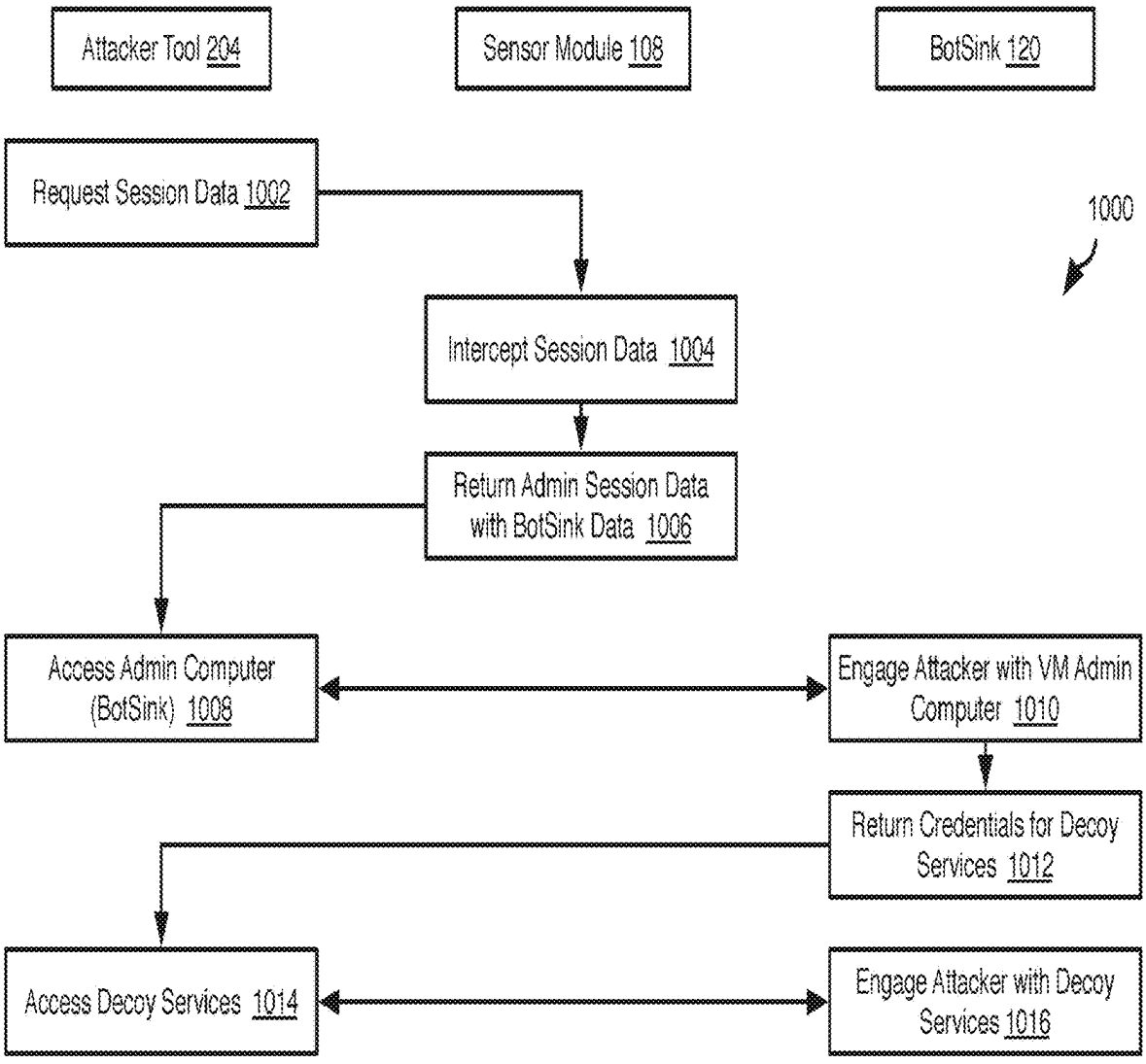

Attacker Tool 204
Sensor Module 108
BotSink 120
Request Session Data 1002
1000
Intercept Session Data 1004
Return Admin Session Data with BotSink Data 1006
Access Admin Computer (BotSink) 1008
Engage Attacker with VM Admin Computer 1010
Return Credentials for Decoy Services 1012
Access Decoy Services 1014
Engage Attacker with Decoy Services 1016

Fig. 10

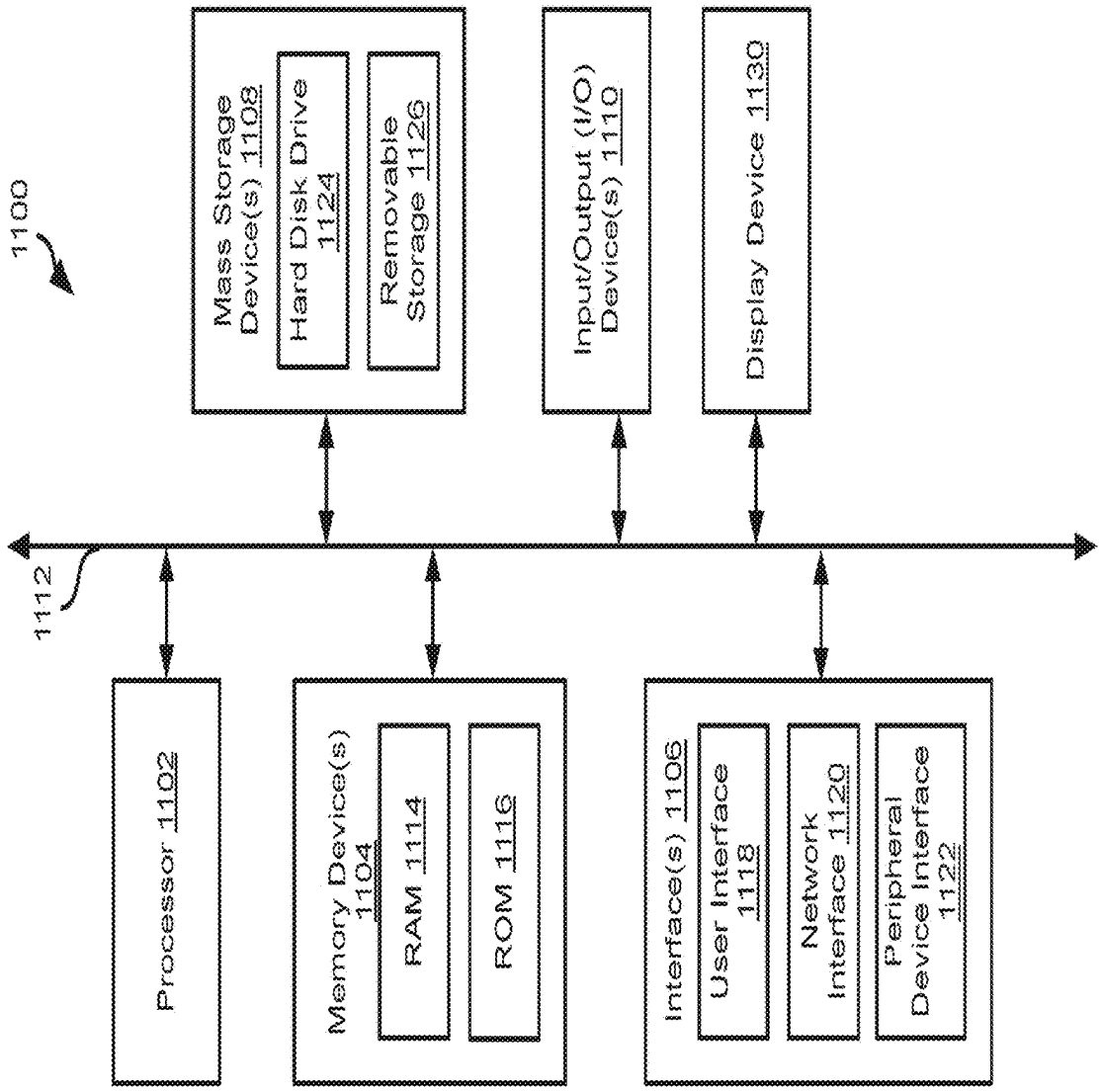

1100
Mass Storage Device(s) 1108
Hard Disk Drive 1124
Removable Storage 1126
Input/Output (I/O) Device(s) 1110
Display Device 1130
1112
Processor 1102
Memory Device(s) 1104
RAM 1114
ROM 1116
Interface(s) 1106
User Interface 1118
Network Interface 1120
Peripheral Device Interface 1122

Fig. 11

DECEIVING ATTACKERS ACCESSING ACTIVE DIRECTORY DATA

RELATED APPLICATIONS

[0001] This application is a continuation of U.S. application Ser. No. 18/173,611, filed Feb. 23, 2023, which is a continuation of U.S. application Ser. No. 16/543,189, filed Aug. 16, 2019, and entitled Title: DECEIVING ATTACKERS ACCESSING ACTIVE DIRECTORY DATA, which is a continuation-in-part of U.S. application Ser. No. 15/383,522, filed Dec. 19, 2016, and entitled DECEIVING ATTACKERS IN ENDPOINT SYSTEMS, which is hereby incorporated herein by reference in its entirety.

BACKGROUND

[0002] Once an end point system is compromised, attackers try to move laterally in the network. Attackers harvest data from end point systems and then use that information to move laterally. The systems and methods disclosed herein provide an improved approach for preventing unauthorized access to application data on endpoint systems.

BRIEF DESCRIPTION OF THE FIGURES

[0003] In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be described and explained with additional specificity and detail through use of the accompanying drawings, in which:

[0004] FIG. **1** is a schematic block diagram of a network environment for performing methods in accordance with an embodiment of the present invention;

[0005] FIG. **2** is a diagram illustrating components for preventing unauthorized access to application data in accordance with an embodiment of the present invention;

[0006] FIGS. **3**A and **3**B are process flow diagrams of methods for preventing unauthorized access in accordance with an embodiment of the present invention;

[0007] FIG. **4** is a schematic block diagram of an alternative network environment for performing methods in accordance with an embodiment of the present invention;

[0008] FIG. **5** is a diagram illustrating components for preventing unauthorized access to active directory data in accordance with an embodiment of the present invention;

[0009] FIG. **6** is a process flow diagram of a method for preventing unauthorized access to active directory data in accordance with an embodiment of the present invention;

[0010] FIG. **7** is a diagram illustrating components for preventing unauthorized access to domain data in accordance with an embodiment of the present invention;

[0011] FIG. **8** is a process flow diagram of a method for preventing unauthorized access to domain data in accordance with an embodiment of the present invention;

[0012] FIG. **9**A is a schematic block diagram illustrating exploitation of an active directory server using an infected endpoint;

[0013] FIG. **9**B is a schematic block diagram illustrating prevention of exploitation of an active directory server using an infected endpoint in accordance with an embodiment of the present invention;

[0014] FIG. **10** is a process flow diagram illustrating the use of decoy session data in accordance with an embodiment of the present invention; and

[0015] FIG. **11** is a schematic block diagram of a computer system suitable for implementing methods in accordance with embodiments of the present invention.

DETAILED DESCRIPTION

[0016] It will be readily understood that the components of the invention, as generally described and illustrated in the Figures herein, could be arranged and designed in a wide variety of different configurations. Thus, the following more detailed description of the embodiments of the invention, as represented in the Figures, is not intended to limit the scope of the invention, as claimed, but is merely representative of certain examples of presently contemplated embodiments in accordance with the invention. The presently described embodiments will be best understood by reference to the drawings, wherein like parts are designated by like numerals throughout.

[0017] Embodiments in accordance with the invention may be embodied as an apparatus, method, or computer program product. Accordingly, the invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.), or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "module" or "system." Furthermore, the invention may take the form of a computer program product embodied in any tangible medium of expression having computer-usable program code embodied in the medium.

[0018] Any combination of one or more computer-usable or computer-readable media may be utilized. For example, a computer-readable medium may include one or more of a portable computer diskette, a hard disk, a random access memory (RAM) device, a read-only memory (ROM) device, an erasable programmable read-only memory (EPROM or Flash memory) device, a portable compact disc read-only memory (CDROM), an optical storage device, and a magnetic storage device. In selected embodiments, a computer-readable medium may comprise any non-transitory medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

[0019] Computer program code for carrying out operations of the invention may be written m any combination of one or more programming languages, including an object-oriented programming language such as Java, Smalltalk, C++, or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages, and may also use descriptive or markup languages such as HTML, XML, JSON, and the like. The program code may execute entirely on a computer system as a stand-alone software package, on a stand-alone hardware unit, partly on a remote computer spaced some distance from the computer, or entirely on a remote computer or server. In the latter scenario, the remote computer may be connected to the computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0020] The invention is described below with reference to flowchart illustrations and/or block diagrams of methods,

apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions or code. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0021] These computer program instructions may also be stored in a non-transitory computer-readable medium that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable medium produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0022] The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0023] Referring to FIG. **1**, the methods disclosed herein may be practiced in a network environment **100** including a plurality of domains **102***a*-**102***c*. The domains **102***a*-**102***c* may be any network division, such as a subnet, local area network (LAN), virtual local area network (VLAN), or the like. The domains **102***a*-**102***c* may be distributed within a same building or over a large geographic area with interconnecting links including the Internet **104**. The illustrated domains **102***a*-**102***c* may represent one or more network components, such as routers, switches, servers, and the like that implement routing of traffic within the domains **102***a*-**102***c* and control traffic flowing into and out of the domains **102***a*-**102***c*.

[0024] Each domain may include one or more endpoints **106***a*-**106***g*. The endpoints **106***a*-**106***g* are production computing devices that operate as personal computers for users or servers providing production services to other endpoints or to external computers accessing the network environment by way of the internet **104**. The endpoints **106***a*-**106***g* may be desktop or laptop computers, mobile phones, tablet computers, server computers, and any other type of computing device. Some endpoints **106***a*-**106***g* may include internet-enabled devices, i.e. so-called internet of things (IoT) devices that are often a vulnerability.

[0025] The endpoints **106***a*-**106***g* are not dedicated honeypots, but rather perform non-decoy functions and process legitimate production data and legitimate production tasks of an enterprise, such as functioning as user computers executing applications such as word processors, browsers, graphics programs etc. The endpoints **106***a*-**106***g* may also function as web servers, database servers, remote login servers, application servers, and the like.

[0026] Some or all of the endpoints **106***a*-**106***g* execute a sensor module **108**. The sensor module **108** stores or accesses a list of sanctioned applications **110***a* and may also store or access a listing or description of protected data **110***b*. The endpoints **106***a*-**106***g* may execute one or more instances of one or more of the sanctioned applications **110***a* and store data generated or used by these applications that corresponds to the protected data. The sanctioned applications **110***a* are production applications for performing any of the production tasks and functionality mentioned above. Accordingly, the protected data **110***b* may also be production data for use in performing the production tasks and functionality mentioned above, as opposed to decoy or deceptive data.

[0027] The sensor module **108** may store or access deception data **110***c* stored locally on the endpoint **106***a*-**106***g* or accessed from another location. The deception data **110***c* may mimic the format of production data in the form of web pages, word processor documents, spreadsheets, databases, etc. The deception data **110***c* may also mimic other files used by applications such as credentials for authenticating the application with a remote server, configuration files, browser histories, a listing of recently accessed files, configuration files, and the like.

[0028] In the case of credentials or other files that are used to access a remote server or provide a record of accessing a remote server, the deception data **110***c* may reference a BotSink **120**. The BotSink **120** may function as a honey pot programmed to engage an attacker while preventing access to production data or computer systems. For example, the BotSink **120** may execute one or more virtual machines implementing network services that engage and monitor malicious code while preventing access to other endpoints **106***a*-**106***g* of the network. The BotSink **120** may implement any of the method methods for detecting and engaging malicious code disclosed in the following applications (herein after "the incorporated applications"), which are hereby incorporated herein by reference in their entirety:

[0029] U.S. application Ser. No. 14/458,026, filed Aug. 12, 2014, and entitled DISTRIBUTED SYSTEM FOR BOT DETECTION;

[0030] U.S. application Ser. No. 14/466,646, filed Aug. 22, 2014, and entitled EVALUATING URLS FOR MALICIOUS CONTENT;

[0031] U.S. application Ser. No. 14/549,112, filed Nov. 20, 2014, and entitled METHOD FOR DIRECTING MALICIOUS ACTIVITY TO A MONITORING SYSTEM;

[0032] U.S. application Ser. No. 15/157,082, filed May 17, 2016, and entitled EMULATING SUCCESSFUL SHELLCODE ATTACKS;

[0033] U.S. application Ser. No. 14/805,202, filed Jul. 21, 2015, and entitled MONITORING ACCESS OF NETWORK DARK.SPACE;

[0034] U.S. application Ser. No. 14/965,574, filed Dec. 10, 2015, and entitled DATABASE DECEPTION IN DIRECTORY SERVICES;

[0035] U.S. application Ser. No. 15/142,860, filed Apr. 29, 2016, and entitled AUTHENTICATION INCIDENT DETECTION AND MANAGEMENT;

[0036] U.S. application Ser. No. 15/153,471, filed May 12, 2016, and entitled LURING ATTACKERS TOWARDS DECEPTION SERVERS;

[0037] U.S. application Ser. No. 15/204,779, filed Jul. 7, 2016, and entitled DETECTING MAN-IN-THE-MIDDLE ATTACKS; and

[0038] U.S. application Ser. No. 15/360,117, filed Nov. 23, 2016, and entitled IMPLEMENTING DECOYS IN NETWORK ENDPOINTS.

[0039] In some embodiments, the data **110***a*-**110***c* is provided to the endpoints **106***a*-**106***g* by a management server **112**. The management server **112** may implement an application policy module **114**. The application policy module **114** stores or accesses a listing **116***a* of sanctioned applications and may provide an interface for an administrator to specific what applications are included in the listing **116***a*. The listing **116***a* may indicate which applications are sanctioned for particular endpoints **106***a*-**106***g* or for endpoints in a particular domain **102***a*-**102***c*. The listing **116***a* may be automatically updated to include applications as they are installed on endpoints **106***a*-**106***g* by an administrator.

[0040] In a similar manner, the management server **112** may store or access protected data **116***b* that lists data files, folders, or other descriptors of data that are protected. The protected data **116***b* may be obtained automatically from configuration files for applications. For example, an application on installation will often create directories for files used by the application. Accordingly, these directories and files may be observed by analyzing the configuration files for instructions to create these directories and files or by observing how the file system changes following installation of the application on a sample endpoint.

[0041] The management server **112** may also store or access deception data **118**. As noted above, this deception data may mimic production data for one or more applications and may reference the BotSink **120** in order to lure attackers into engagement with the BotSink **120**.

[0042] The management server **112** distributes some or all of the data **116***a*-**116***b*, **118** to the endpoints **106***a*-**106***g*. For example, deception data **118** provided to an endpoint may mimic files generated or used by one or more production applications actually installed on that endpoint. Likewise, the listing of sanctioned applications **116***a* provide to the endpoint may include only those applications that were installed by an administrator or authorized user on that endpoint. The deception data **118** provided to the endpoint may then include deception data mimicking the production data of those applications. The management server **112** may periodically update the data **116***a*-**116***b*, **118** and distribute updated data to the endpoints **106***a*-**106***g*.

[0043] The methods disclosed herein are invoked in response to an attacker system **122** attempting to access production application data on an endpoint **106***a*-**106***g*. This may include the attacker system **122** issuing commands to the endpoint **106***a*-**106***g* or uploading malicious code to the endpoint **106***a*-**106***g*, which then attempts to access the production application data. However, unauthorized access may be prevented using the methods disclosed herein in either case.

[0044] Referring to FIG. **2**, the sensor module **108** may be incorporated into the operating system **200**, such as by modifying the functions of one or more libraries for performing access to a file system. The sensor module **108** intercepts file system command from instances of applications **202** installed on the endpoint as well as file system commands from an attacker, such as an attacker tool **204** executing on the endpoint. The file system commands that are intercepted may include write commands, read commands, delete comments, or instructions to list the contents of a directory or other commands to navigate through a directory.

[0045] The sensor module **108** evaluates a source of each command, e.g. the binary instance executing on the endpoint that issued the command to the operating system. If the source of the command is an application listed in the sanctioned applications **110***a*, the command is passed to the file system I/O (input output) functions **206** of the operating system **200**, which then executes the command with respect to the production application data **208**.

[0046] If the source of the command is not found to be in the sanctioned applications **110***a*, the command may be modified such that it refers to deception data **110***c*. The modified command may then be input to the file system I/O functions **206**. In the case of a write or delete command, the sensor module **108** may suppress execution of the command and return an acknowledgment to the source of the command indicating that the command was executed successfully.

[0047] In the embodiment of FIG. **2**, the modified commands are executed by the same file system I/O functions **206** as other commands. Accordingly, the deception data **110***c* may be stored in the file system and accessible to such functions **206**. However, the deception data **110***c* may be hidden such that it is not viewable by users or applications executing on the endpoint.

[0048] In other embodiments, the sensor module **108** itself may access and return the deception data **110***c*, in which case the deception data **110***c* may be stored anywhere, including remotely from the endpoint and referenced by the sensor module **108**. The deception data **110***c* could, for example, be encrypted such that only the sensor module **108** can decrypt the deception data **110***c* in order to return it in response to a read command from the attacker tool **204** or other unauthorized source.

[0049] In still other embodiments, the sensor module **108** may automatically generate deception data **110***c* in response to file system commands, such as based on templates that are populated with random data in order to mimic a type of file requested by the file system command from the attacker tool **204**.

[0050] Referring to FIG. **3**A, the sensor module **108** may include any executable code programmed to execute the illustrated method **300***a*. The method **300***a* may include receiving **302** a file system command, such as by intercepting a command made to the operating system **200** of the endpoint in which the sensor module **108** is embedded.

[0051] The method **300***a* may include evaluating **304** whether the data (file, directory, type of file, etc.) is protected data, such as might be indicated by the listing **110***b* of protected data on the endpoint. If not, the method **300***a* may include executing **306** the file system command without modification, i.e. passing it to the operating system **200** for execution. In some embodiments, only the sanctioned applications **110***a* are permitted to issue file system commands, which may include operating system utilities. Accordingly, in some embodiments, step **304** may be omitted.

[0052] The method **300***a* may further include evaluating a source of the file system command according to some or all of steps **308**-**312**. For example, if the source of the file system command is found **308** to have a certificate matching

that of a sanctioned application **110***a*, then file system command may be executed **306** with respect to the data referenced in the command.

[0053] If a hash, e.g. the SHA-1 (secure hash algorithm) hash of the binary code that issued the file system command is found **310** to match a hash of the binary executable for one of the sanctioned applications **110***a*, then the file system command may be executed **306** with respect to the data referenced in the command.

[0054] If a path to the binary code that issued the file system command is found **312** to match the path to the binary executable of one of the sanctioned applications **110***a*, then the file system command may be executed **306** with respect to the data referenced in the command.

[0055] Steps **308-312** are just examples of checks that may be used to verify whether binary code issuing a command is in fact an instance of a sanctioned application. In some embodiments, all of steps **308-312** must be satisfied before step **306** will be executed with respect to the data referenced in the command. In other embodiments, other checks may be used as alternatives or as additional requirements before step **306** will be executed with respect to the data referenced in the command.

[0056] In some embodiments, certain protected data may be bound to a particular sanctioned application **110***a*. Accordingly steps **308-312** may be evaluated only for those applications that are bound to the protected data **110***b* referenced in the file system command, i.e. the command will be executed with respect to the data referenced in the command only if one of **308-312** (or each and every one of **308-312** in some embodiments) is satisfied for at least one sanctioned application **110***a* that is also bound to the protected data **110***b* referenced in the file system command.

[0057] If the tests of steps **308-312** are not sufficient to invoke execution of step **306** for the data referenced in the file system command according to any of the embodiments mentioned above, then the source of the file system command may be determined not to be one of the sanctioned applications **110***a* and processing continues at steps **314**, **318**.

[0058] If the file system command is found **314** to be a read comment, then file system command may be changed **318** to refer to the deception data **110***c* of the endpoint. Step **306** may be executed with respect to the modified file system command, which will include returning the deception data referenced by the modified file to the source of the file system command, such as to the attacker tool **204**.

[0059] If the file system command is found **316** not to be a read command, such as in the case of a write or a delete command, then the file system command is ignored and a simulated acknowledgment of the command is returned **316** to the source of the file system command.

[0060] Referring to FIG. **3**B, in some embodiments, the sensor module **108**, or a different sensor module **108** may execute the illustrated method **300***b* with respect to commands other than file system commands. In some embodiments, a plurality of sensor modules **108** execute on the endpoint and each sensor module **108** may intercept a different types of command. Each sensor module will then execute the method **300***a* or **300***b* upon intercepting that type of command. For example, requests to modify, delete, or read information regarding processes executing on the endpoint, the registry of the endpoint, or an application programming interface (API) available to applications executing on the endpoint. The method **300***b* may include receiving **320** a command and evaluating **322** whether the command references protected data or is a protected command. If not, then the command may be executed **324**. For example, the protected data **110***b* may indicate which commands are restricted to sanctioned applications **110***a* in addition to data or types of data that are protected. In some embodiments, only sanctioned applications are permitted to access any data or invoke execution of any command. Accordingly, step **322** may be omitted as a path to execution **324** of a command.

[0061] The method **300***b* may further include evaluating the source of the command of step **320** according to steps **308-312** in the same manner as for the method **300***a*. If the conditions of steps **308-312** are met, then the command may be executed in the same form it was received. As for the method **300***a*, all of steps **308-312** must have a positive result before the command is executed **324** and one or more additional tests may be required to be satisfied before the command is executed **324** without modification.

[0062] If the steps **308-312** indicate that the source is a sanctioned application, then the method **300***b* may include evaluating **326** whether the command is a request for information, such as a request for information regarding executing processes, the registry, available APIs, or other system information. If so, then deception data is returned **328** to the source of the command. As for other embodiments disclosed herein, the deception data that is returned may mimic the legitimate data that is requested but not correspond to actual system data.

[0063] If the command is not a request for information, the method **300***b* may include returning **330** a result that simulates successful execution of the command. As for the method **300***a*, a delete command may indicate that data was successfully deleted, a modify command may indicate that the system data or process or operating parameter referenced in the command was modified as requested.

[0064] An example of the use of the method **300***a* may include the browser history for a browser. For example, the FIREFOX browser history may be stored at C:\Users\<user name>\AppData\Roaming\Mozilla Firefox\Profiles\<some profile number>.default\formhistory.sqlite. Ordinarily, only the FIREFOX browser should access this file inasmuch as it will include functions for displaying the browser history on request.

[0065] Accordingly, any other application attempting to access this file or its directory may be blocked by the sensor module **108** and instead receive a simulated browser history file including decoy data.

[0066] In another example, in Windows, the "shawdow-copy delete" command deletes the volume shadow copies. Malware often deletes this to accessing of backup volumes.

[0067] Accordingly, this file may be listed as protected data **110***b* and sensor module **108** will detect attempts to delete the volume shadow copies by non-sanctioned applications and prevent their execution. As noted above, acknowledgments of such commands may be returned indicating that the volume shadow copies were in fact deleted.

[0068] In an example of the use of the method **30***b*, an attacker, e.g. attacker tool **122**, tries to access the registry, which may be in a listing of protected data **110***b*. For example, the attacker may attempt to read the registry to determine a registry key for one or more antivirus application (e.g., MCAFEE, SYMANTEC, or the like) is present in the registry. No legitimate application would generally need

to access the registry to determine whether an antivirus application is installed. In some embodiments, the sensor module **108** may intercept such attempts and return a result including a registry key for the antivirus tool, regardless of whether the antivirus application is installed. In many cases, this will cause the attacker tool to refrain from installing itself or otherwise attempting to perform malicious activities.

[0069] In another example, the attacker, e.g. attacker tool **122**, seeks to determine whether the endpoint is a virtual machine (VM), such as by evaluating the registry to determine whether a hypervisor is installed and/or executing on the endpoint. The sensor module **108** intercepts these requests and returns an output indicating that the endpoint is executing a VM,

[0070] e.g. indicate that the current operating environment in which the command was received is a VM or that a hypervisor is installed and/or executing on the endpoint. In another example, the attacker, e.g. attacker tool **122**, seeks to view a list of processes executing on the endpoint. In response, the sensor module **108** will return a "correct list of processes list" to the attacker.

[0071] Referring to FIG. **4**, the network environment **100** may further include an active directory server **402** in one or more of the domains **102***a*-**102***c*. The active directory server **402** may implement a directory service. A directory service functions as databases that map and store the names of network resources to their respective network addresses. Users referencing network objects need not remember the physical address of the object. The directory may store network resources and those resources may or may not have a name. The directory can also store user identifiers (very common), departments of user identifiers, access level of user identifiers, hosts accessible by user identifiers, the access level associated with each user identifier. The directory may further store organizational details, a network topology, an IT policy, and the like. Although Active Directory, the most popular directory server, depends on DNS very heavily, it is distinct therefrom and provides much more than a simple mapping between domain names and IP addresses. In particular, directory services may perform authentication for access to resources as well. LDAP (Lightweight Directory Access Protocol) is one of the popular methods available to access the data in directory services. LDAP also provides authentication and authorization to let user access resources in directory services.

[0072] The directory service implemented by the active directory server **402** may provide authorization and access to key assets in corporate networks. Attackers may use various methods of exploitation to get unauthorized access to directory services. Once an attacker obtains access to a directory service, the attacker can easily log into key servers, databases etc. by impersonating credentials stored in directory services. The attacker may then exfiltrate data. For example, confidential data may be disclosed. In some instances, unauthorized modifications could be made (such as new account creation, access control modifications, document forgery, backup corruption, unauthorized financial transactions etc.) or data may be made unavailable (such as crypto malware, bank account takeover, bringing down or defacement of corporate web servers).

[0073] The systems and methods disclosed herein incorporate database deception into directory services so that attackers cannot differentiate between real production assets with respect to fake assets. Once an attacker access fake credentials pointing to a BotSink **120**, the system prevents outbound access and simulates access to actual enterprise assets. This enables the identification and analysis of attackers for use in preventing data breaches.

[0074] "Active Directory" (AD) is one of the directory services supported by WINDOWS operating systems. The AD domain controller is a server that provides authentication services within a domain whereby it provides access to computer resources. AD provides a database storing information about objects. Each object can be a user, computer, or a group of users. Although the systems and methods disclosed herein are described for use with an Active Directory system, they may be applied with equal effectiveness to any directory service.

[0075] The systems and methods described herein returns simulated responses to requests to an active directory server **402**. These responses reference the BotSink **120** and may include credentials for authenticating with respect to the BotSink **120**, such as a user account defined on the BotSink **120** or a service executing on the BotSink **120**. These responses lure an attacker to use the service or to attempt to use the user account. However, use of such services or accounts results in engagement of the attacker with the BotSink **120** and other resources described herein. This enables early detection of malicious intent and account misuse and diverts attention from real targets and resources.

[0076] Referring to FIG. **5**, in some embodiments, the sensor module **108** on an endpoint **106***a*-**106***g* may detect responses from an active directory server **402**. These responses may be the result of requests issued to the active directory server **402** by an application **202** or attacker tool **204** executing on that endpoint **106***a*-**106***g*. For example, the sensor module **108** may operate in conjunction with substituted or modified functions in an active directory API (application programming interface) in the operating system **200** of an endpoint **106***a*-**106***g*. The substituted or modified functions may route responses from the active directory server **402** to the sensor module **108** prior to return of the response to the application that requested it. The sensor module **108** may then determine whether to return the response to an application **202** to which the response is addressed without modification or with modification. In particular, if the response is addressed to an attacker tool **204** that is not a sanctioned application, the response may be modified such that any system, user account, or credential included in the response is replaced with corresponding references to the BotSink **120**, decoy computer on the BotSink **120**, a credential or user account for authenticating with the BotSink **120** (e.g., logging in, accessing a particular service, etc.).

[0077] The manner in which the application to which the response is addressed is determined to be sanctioned or not may be according to the approach described in either of FIGS. **3**A and **3**B. In particular, the response from the active directory server **402** may be received on a port with respect to which an application is registered by the operating system **200**.

[0078] Accordingly, the sensor module **108** may evaluate that application to determine whether it is sanctioned according to the approach of either of FIGS. **3**A and **3**B. If not, the response is modified at step **606** to replace references to addresses, actual user accounts, services, computer systems, or other production resources with corresponding

references to the BotSink **120**, i.e. an address of the BotSink **120**, a decoy user account, decoy service, decoy virtual machine, or other decoy resource executing on the BotSink **120**.

[0079] Referring to FIG. **6**, the sensor module **108** may execute the illustrated method **600** with respect to responses from the active directory server **402**. The method **600** includes receiving **602** a response from the active directory server **402** by the sensor module **108**. The sensor module **108** evaluates **604** whether the application to which the response is addressed is sanctioned. For example, this may include evaluating an application listening to a port to which the response was addressed. Alternatively, the response may include some other identifier that is sufficient to identify the application (i.e. instance of an application executing on operating system **200** to which the response is addressed. As noted above, receiving the response may be performed by interfacing with an API programmed to interface with the active directory server **402** such that response are intercepted before they are returned to the requesting application. For example, the sensor module **108** may interface with substituted or modified versions of the Power Shell such that commands input to the Power Shell are intercepted and possibly modified by the sensor module **108**. Commands that may be intercepted may include commands such as 'net group/domain "domain users"' will return list of domain users present in Active Directory. Other example commands include 'Get-ADComputer-Filter*' will return all computer accounts in Active Directory, "nltest/dclist: domain name" will return domains in the network.

[0080] The modified response is then returned **608** to the application to which it is addressed, i.e. the application that requested the information included in the response from the active directory server **402**. Where the application is found **604** to be sanctioned, the original response, or data from the response, is returned **608** to the source of the request without substituting references to the BotSink **120**.

[0081] Note that FIG. **6** illustrates the case where a response from the active directory service is received and modified. In other embodiments, a request from an application that is not sanctioned that is addressed to the active directory service is intercepted and simulated response is generated that references the BotSink **120**, i.e. a response that has an expected format corresponding to the request but with names of systems, addresses, credentials, services, etc. referencing such entities as implemented by the BotSink **120**.

[0082] In some embodiments, the active directory server **402** may host decoy information describing a configuration of the BotSink **120**, such as decoy information describing decoy users accounts, addresses of decoy systems (virtual machines), decoy services, and decoy credentials for accessing any of these accounts, systems, and services. Accordingly, modifying a request by the sensor module may include modifying the request such that it is a request for decoy information on the active directory server (e.g., references the BotSink **120**) such that the response from the active directory server **402** includes only the decoy information rather than the information actually requested.

[0083] In some embodiments, authenticated users of an endpoint **106***a*-**106***g* are added to a whitelist accessed by the sensor module **108** such that requests to the active directory server **402** and responses received are not substituted according to the method **600**.

[0084] Referring to FIG. **7**, in some embodiments, the sensor module **108** intercepts responses to requests to a domain controller **700**. In some embodiments, the function of the domain controller **700** is performed by a domain joined computer. For example, the sensor module **108** may monitor one or both of requests to a network API **702** of the operating system **200** to the domain controller **700** and responses to such requests.

[0085] For example, referring to the method **800** of FIG. **8**, the responses may be responses to requests for session data issued received **802** from a network API **702**. A response determined **804** to be to a non-sanctioned application may be modified at step **806** to refer to the BotSink **120** and returned **808** to the non-sanctioned application. In one scenario, an attacker tool **204** executing on an infected endpoint (e.g., endpoint **106***a*) requests session data from the domain controller **700** or domain joined computer **700** using information learned via earlier active directory queries.

[0086] Responses determined **804** to be in response to requests from sanctioned applications **202** may be returned **808** to them without replacing a reference to a computer system in the response with a reference to the BotSink **120**. The manner in which an application is determined **804** to be sanctioned may be performed in the same manner described above with respect to either of FIGS. **3**A and **3**B.

[0087] FIGS. **9**A and **9**B illustrate how the methods **600** and **800** may be used to redirect and occupy an attacker. In an unprotected case shown in FIG. **9**A, an attacker tool **204** on an infected endpoint **106***a* uses the active directory API on the infected system to access an active directory database **900** through the active directory server **402**. Using this information, the attacker tool **204** uses the network API to request session data in order to identity of an endpoint **106***b* logged in to the domain account for a domain to which infected endpoint **106***a* belong. The attacker tool may infect the administrator endpoint **106***b* and access the active directory database **900** and discover that the admin user account is referenced in domain data **904** defining another group, e.g. an administrator group. The endpoint **106***b* may further store credentials enabling the endpoint **106***b* to access a service hosted by server **106***f*.

[0088] Accordingly, the attacker tool **204** on the infected system **106***a* may attempt to move laterally by requesting information from the active directory server **402** in order to discover the endpoint **106***b* of the admin user account, infect it, and use it to access the service on server **106***f* as an administrator.

[0089] FIG. **9**B illustrates an alternative view seen by an attacker tool **204** on the infected endpoint **106***a* when the methods according to FIGS. **4** through **6** are implemented. The attacker tool **204** requests information from the active directory server **402**. The responses to these requests are modified to reference decoy information, such as decoy active directory data **906** having a same format as the active directory database **900** but with references to computer systems and services being replaced with references to the BotSink **120** (e.g. virtual machines executing on the BotSink **120**) and services executing on the BotSink **120**.

[0090] A response to a request to obtain the identity of an administrator of a domain to which the endpoint **106***a* belongs using the network API is intercepted and modified to reference to a decoy domain **908**. A request for the identity of the administrator of the decoy domain **908** may include a reference to a user account **912** or virtual machine logged in

to a user account **912** on the BotSink **120**. That user account **912** may be defined as part of an admin domain **910** and that user account **912** may include credentials (e.g., VM for which the user account **912** is logged in may store or reference credentials) that are sufficient to authenticate a user with respect to a service **914** executing on the BotSink **120**.

[0091] Accordingly, the attacker tool **204** may attempt to move laterally as in the case of FIG. **9**A but only engage the BotSink **120**. The activities of the attacker tool **204** may be monitored and logged by the BotSink **120** in order to characterize the attacker tool **204** and alerts may be generated to alert and administrator to remove the attacker tool **204** from the infected endpoint **106***a*. Engaging, monitoring, logging, characterizing, and generation of alerts may be performed according to any of the approaches described in the incorporated applications.

[0092] FIG. **10** illustrates a method **1000** that may be implemented using the system shown in FIG. **9**B. An attacker tool **204** executing on an infected endpoint (e.g., endpoint **106***a*) requests **1002** session data using a network API for the domain of the infected endpoint, the session data indicating computers (e.g., computers in the same domain) that are connected to the active directory server **402**. For example, a network API command for requesting such session data may be used. The sensor module **108** intercepts **1004** a response to this request and determines that the attacker tool **204** is a non-sanctioned application attempting to access restricted data (see FIG. **6**). In response, the sensor module **108** returns **1006** the session data with references to one or more computers in the session data replaced with one or more addresses assigned to the BotSink **120**. In the alternative, the sensor module **108** intercepts the request for session data and creates a decoy response referencing the BotSink **120** rather than modifying a response from the active directory server **402**.

[0093] In particular, the IP address of the computer logged in to the administrator account of the domain may be replaced with an IP address assigned to the BotSink **120**. For example, the BotSink **120** may acquire IP addresses in various domains as described in the incorporated applications. Each address added to the session data may be assigned to a virtual machine (VM) executing on the BotSink **120**.

[0094] The attacker tool **204** may then attempt to access **1008** the computer logged in to the administrator account, which is in fact the BotSink **120**. The BotSink **120** (e.g., the VM assigned the IP address represented as that of the administrator's computer at step **1006** ("the admin VM")) may then engage **1010** the attacker tool. The activities of the attacker tool **204** with respect to the BotSink **120** may be monitored and stored in order to characterize the attacker tool **204** as described in the incorporated applications.

[0095] In particular, the attacker tool **204** may attempt to obtain credentials or identifiers of services cached or stored by the admin VM. In response, these credentials may be returned **1012** to the attacker tool. However, these credentials or identifiers of services may reference services implemented by the BotSink **120**, such as by other VMs being executed by the BotSink **120**. The attacker tool **204** receives the credentials or identifiers and attempts to access **1014** the services represented by them.

[0096] In response, the BotSink **120** authenticates the attacker tool **204** and engages **1016** with the attacker tool **204** using the service, i.e. executes commands from the attacker tool in accordance with the service. Actions of the attacker tool **204** may be monitored and used to characterize the attacker tool **204** as described in the incorporated applications. In addition or as an alternative, interaction with the BotSink **120** at steps **1010**, **1012**, and **1016** by the attacker tool may trigger an alert or remedial action such as removing the attacker tool **204** from the infected endpoint **106***a* or isolating (disconnecting) the infected endpoint **106***a* from a network.

[0097] Note that in some embodiments, one of the services implemented on the BotSink **120** may be a decoy active directory service that implements an interface for responding to commands in the active directory API and is programmed to response to these commands with decoy data referencing decoy computers, user accounts, and services implemented on the BotSink **120**.

[0098] FIG. **11** is a block diagram illustrating an example computing device **1100** which can be used to implement the system and methods disclosed herein. The endpoints **106***a*-**106***g*, management server **112**, BotSink **120**, attacker system **122**, and active directory server **402** may also have some or all of the attributes of the computing device **1100**. In some embodiments, a cluster of computing devices interconnected by a network may be used to implement any one or more components of the invention.

[0099] Computing device **1100** may be used to perform various procedures, such as those discussed herein. Computing device **1100** can function as a server, a client, or any other computing entity. Computing device can perform various monitoring functions as discussed herein, and can execute one or more application programs, such as the application programs described herein. Computing device **1100** can be any of a wide variety of computing devices, such as a desktop computer, a notebook computer, a server computer, a handheld computer, tablet computer and the like.

[0100] Computing device **1100** includes one or more processor(s) **1102**, one or more memory device(s) **1104**, one or more interface(s) **1106**, one or more mass storage device(s) **1108**, one or more Input/Output (I/O) device(s) **1110**, and a display device **1130** all of which are coupled to a bus **1112**. Processor(s) **1102** include one or more processors or controllers that execute instructions stored in memory device(s) **1104** and/or mass storage device(s) **1108**. Processor(s) **1102** may also include various types of computer-readable media, such as cache memory.

[0101] Memory device(s) **1104** include various computer-readable media, such as volatile memory (e.g., random access memory (RAM) **1114**) and/or nonvolatile memory (e.g., read-only memory (ROM) **1116**). Memory device(s) **1104** may also include rewritable ROM, such as Flash memory.

[0102] Mass storage device(s) **1108** include various computer readable media, such as magnetic tapes, magnetic disks, optical disks, solid-state memory (e.g., Flash memory), and so forth. As shown in FIG. **11**, a particular mass storage device is a hard disk drive **1124**. Various drives may also be included in mass storage device(s) **1108** to enable reading from and/or writing to the various computer readable media. Mass storage device(s) **1108** include removable media **1126** and/or non-removable media.

[0103] I/O device(s) **1110** include various devices that allow data and/or other information to be input to or

retrieved from computing device **1100**. Example I/O device (s) **1110** include cursor control devices, keyboards, keypads, microphones, monitors or other display devices, speakers, printers, network interface cards, modems, lenses, CCDs or other image capture devices, and the like.

[0104] Display device **1130** includes any type of device capable of displaying information to one or more users of computing device **1100**. Examples of display device **1130** include a monitor, display terminal, video projection device, and the like.

[0105] Interface(s) **1106** include various interfaces that allow computing device **1100** to interact with other systems, devices, or computing environments. Example interface(s) **1106** include any number of different network interfaces **1120**, such as interfaces to local area networks (LANs), wide area networks (WANs), wireless networks, and the Internet. Other interface(s) include user interface **1118** and peripheral device interface **1122**. The interface(s) **1106** may also include one or more user interface elements **1118**. The interface(s) **1106** may also include one or more peripheral interfaces such as interfaces for printers, pointing devices (mice, track pad, etc.), keyboards, and the like.

[0106] Bus **1112** allows processor(s) **1102**, memory device (s) **1104**, interface(s) **1106**, mass storage device(s) **1108**, and I/O device(s) **1110** to communicate with one another, as well as other devices or components coupled to bus **1112**. Bus **1112** represents one or more of several types of bus structures, such as a system bus, PCI bus, IEEE 1394 bus, USB bus, and so forth.

[0107] For purposes of illustration, programs and other executable program components are shown herein as discrete blocks, although it is understood that such programs and components may reside at various times in different storage components of computing device **1100**, and are executed by processor(s) **1102**. Alternatively, the systems and procedures described herein can be implemented in hardware, or a combination of hardware, software, and/or firmware. For example, one or more application specific integrated circuits (ASICs) can be programmed to carry out one or more of the systems and procedures described herein.

1. (canceled)

2. A method comprising:

configuring, by a computer system, a call to refer to a detour function;

receiving, by the computer system, the call to the detour function from a source to obtain data regarding a network resource;

determining, by the computer system via the detour function, that the source is not on a list of sanctioned applications; and

in response to determining that the source is not on the list of sanctioned applications, returning, by the computer system via the detour function, a response to the call having the data regarding the network resource replaced with data regarding a decoy server.

3. The method of claim **2**, wherein receiving the call comprises receiving a call to a first function, a reference to the first function being substituted for a reference to a second function in a dynamic link library, the first function referencing the second function.

4. The method of claim **3**, wherein the call received from the source is a first system call and the source is a first source, the method further comprising:

receiving, by the computer system, a second call to the first function from a second source;

determining, by the first function, that the second source is on the list of sanctioned applications; and

in response to determining that the second source is on the list of sanctioned applications, invoking, by the first function, the second function.

5. The method of claim **3**, further comprising:

invoking, by the first function, the second function in response to receiving the call from the source;

receiving, by the first function, a result from the second function;

modifying, by the first function, the result to obtain a modified result referencing the decoy server;

replacing, by the first function, a first reference in the response with a second reference referencing a decoy server to obtain a modified response;

returning, by the first function, the modified response to the source.

6. The method of claim **3**, further comprising:

modifying, by the first function, an argument of the call to replace a domain name service (DNS) address with an internet protocol (IP) address of the decoy server to obtain a modified argument;

passing, by the first function, the call with the modified argument to the second function for invocation;

receiving, by the first function, a result from the second function; and

returning, by the first function, the result to the source of the call.

7. The method of claim **2**, wherein the call is an instruction to list network shares mounted to the computer system.

8. The method of claim **2**, wherein the call is an instruction to list credentials for network services stored on the computer system.

9. The method of claim **2**, wherein the call is an instruction to list domain controllers.

10. The method of claim **2**, wherein the call is an instruction to enumerate network computers.

11. The method of claim **2**, wherein the call is an instruction to list users and groups, the method further comprising:

generating, by the computer system, a first response including references to a decoy group defined on the decoy server;

returning, by the computer system, the first response to the source;

receiving, by the computer system from the source, a request for data regarding the decoy group;

in response to the request for data regarding the decoy group, returning a second response to the source, the second response including decoy account information for a decoy user in the decoy group.

12. A system comprising:

a computer system including one or more processing devices and one or more memory devices operably coupled to the one or more processing devices, the one or more memory devices storing executable code that, when executed by the one or more processing devices, causes the one or more processing devices to:

configure a call to refer to a detour function;

receive the call to the detour function from a source to obtain data regarding a remote network resource;

determine, via the detour function, that the source is not on a list of sanctioned applications; and

in response to determining that the source is not on the list of sanctioned applications, return, via the detour function, a response to the call having the information regarding the remote network resource replaced with data regarding a decoy server.

13. The system of claim **12**, wherein the executable code, when executed by the one or more processing devices, further causes the one or more processing devices to receive the call by receiving a call to a first function, a reference to the first function being substituted for a reference to a second function in a dynamic link library, the first function referencing the second function.

14. The system of claim **13**, wherein the call received from the source is a first call and the source is a first source;

wherein the executable code, when executed by the one or more processing devices, further causes the one or more processing devices to:

receive a second call to the first function from a second source;

determine that the second source is on the list of sanctioned applications; and

in response to determining that the second source is on the list of sanctioned applications, invoke, by the first function, the second function.

15. The system of claim **13**, wherein the executable code, when executed by the one or more processing devices, further causes the one or more processing devices to:

invoke, by the first function, the second function in response to receiving the call from the source;

receive, by the first function, a result from the second function;

modify, by the first function, the result to obtain a modified result referencing the decoy server;

replace, by the first function, a first reference in the response with a second reference referencing a decoy server to obtain a modified response;

return, by the first function, the modified response to the source.

16. The system of claim **13**, wherein the executable code, when executed by the one or more processing devices, further causes the one or more processing devices to:

modify an argument of the call to replace a domain name service (DNS) address with an internet protocol (IP) address of the decoy server to obtain a modified argument;

pass, by the first function, the call with the modified argument to the second function for invocation;

receive, by the first function, a result from the second function; and

return, by the first function, the result to the source of the call.

17. The system of claim **12**, wherein the call is an instruction to list network shares mounted to the computer system.

18. The system of claim **12**, wherein the call is an instruction to list credentials for network services stored on the computer system.

19. The system of claim **12**, wherein the call is an instruction to list domain controllers.

20. The system of claim **12**, wherein the call is an instruction to enumerate network computers.

21. The system of claim **12**, wherein the call is an instruction to list users and groups;

wherein the executable code, when executed by the one or more processing devices, further causes the one or more processing devices to:

generate a first response including references to a decoy group defined on the decoy server;

return the first response to the source;

receive, from the source, a request for information regarding the decoy group;

in response to the request for information regarding the decoy group, return a second response to the source, the second response including decoy account information for a decoy user in the decoy group.

* * * * *