(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2018/027823

(22) International Filing Date:
16 April 2018 (16.04.2018)

(25) Filing Language:
English

(26) Publication Language:
English

(30) Priority Data:


Published: without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) Title: METHODS, SYSTEMS, AND MEDIA FOR TESTING INSIDER THREAT DETECTION SYSTEMS

(57) Abstract: Methods, systems, and media for testing insider threat detection systems are provided. In some embodiments, the method comprises: receiving, using a hardware processor, a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts; generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts; selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type; generating a simulated user bot based on the selected model of user behavior; executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment; determining whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and transmitting a notification indicating an efficacy of the insider threat detection system based on the determination.
METHODS, SYSTEMS, AND MEDIA FOR TESTING INSIDER THREAT DETECTION SYSTEMS

Statement Regarding Federally Sponsored Research or Development
[0001] This invention was made with government support under Grant No. P010185636 awarded by the Intelligence Advanced Research Projects Activity (IARPA) of the Office of the Director of National Intelligence. The government has certain rights in the invention.

Technical Field
[0002] The disclosed subject matter relates to methods, systems, and media for testing insider threat detection systems. More particularly, the disclosed subject matter relates to generating and/or injecting simulated user bots into a computing environment to perform security and/or verification tasks.

Background
[0003] Computer viruses, worms, trojans, hackers, rootkits, spyware, key recovery attacks, denial-of-service attacks, malicious software (or malware), probes, etc. are a constant menace to all users of computers connected to public computer networks (such as the Internet) and/or private networks (such as corporate computer networks). Because of these threats, many computers are protected by antivirus software and firewalls. However, these preventative measures are not always adequate. For example, documents can be embedded with malware (e.g., network sniffers, keystroke loggers, etc.) and inserted into a network or a system through a universal serial bus (USB) drive, a compact disk (CD), or downloaded from a reputable source, thereby bypassing preventative measures like firewalls and packet filters.
[0004] In addition, the removal of malware is not always possible. In many situations, malicious code can sometimes be identified, but because the infected system performs a mission-critical operation, the infection is allowed to continue since the system cannot be stopped due to the operational need to keep the system running or the requirement to keep the data intact. This has made filtering-based prevention mechanisms an ineffective and an insufficient defense. In
failing to stop the malware, malware can reach systems in a network and cause serious damage, particularly if the malware is left undetected for long periods of time.

[0005] Using malware or other threats, attackers can snoop or eavesdrop on a computer or a network, download and exfiltrate data, steal assets and information, destroy critical assets and information, and/or modify information. Moreover, it should also be noted that these filtering-based prevention mechanisms are equally ineffective against inside attackers (e.g., human agents or their system, software proxies, etc.). Similar to attackers that gain access to a computer through malware, inside attackers can perform unauthorized activities, such as exfiltrate data, steal assets, destroy critical information, and/or modify information. This has become one of the most serious threats encountered in modern organizations.

[0006] With the advent of insider threat detection systems for detecting inside attackers or insider threats, there is a need in the art for approaches that test or evaluate such detection systems.

[0007] Accordingly, it is desirable to provide methods, systems, and media for testing insider threat detection systems. For example, insider threat detection systems and other detection systems can be tested with the deployment of simulated user bots that can emulate the actions of actual users. In another example, these simulated user bots can be used in a computing environment to perform one or more security and/or verification tasks.

Summary

[0008] Methods, systems, and media for testing insider threat detection systems are provided.

[0009] In some embodiments of the disclosed subject matter, a method for testing insider threat detection systems is provided, the method comprising: receiving, using a hardware processor, a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts; generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts; selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type; generating a simulated user bot based on the selected model of user behavior; executing the simulated user bot in the computing environment, wherein the simulated
user bot injects a second plurality of actions in the computing environment; determining whether
an insider threat detection system executing within the computing environment identifies the
simulated user bot as a malicious user; and transmitting a notification indicating an efficacy of
the insider threat detection system based on the determination.

[0010] In some embodiments, each of the first plurality of actions includes a user
identifier associated with one of the plurality of user accounts, an event, and a timestamp for the
event.

[0011] In some embodiments, each of the first plurality of actions includes a description
of the event. In some embodiments, the description of the event includes at least one of search
terms and page visitation information.

[0012] In some embodiments, at least one of the first plurality of actions includes a user
selection of one or more particular documents in the computing environment.

[0013] In some embodiments, the method further comprises classifying each of the first
plurality of user actions into a category of a plurality of categories of user actions.

[0014] In some embodiments, the method further comprises: storing the first plurality of
actions in a storage device; translating the first plurality of actions in the storage device by
applying a plurality of rules; and generating an action file for each of the plurality of user
accounts based on the translated plurality of actions.

[0015] In some embodiments, the method further comprises modifying the action file by
removing at least one of the first plurality of actions.

[0016] In some embodiments, the method further comprises: deriving user parameters
based on the translated plurality of actions; and generating an action file based on the derived
user parameters.

[0017] In some embodiments, the user parameters are derived from a subset of actions
that are randomly selected from the translated plurality of actions.

[0018] In some embodiments, the method further comprises: creating a user account in
the computing environment; and introducing the simulated user bot into the computing
environment using the created user account.

[0019] In some embodiments, the method further comprises analyzing the second
plurality of actions by determining whether the insider threat detection system identifies the
simulated user bot as the malicious user.
[0020] In some embodiments, the insider threat detection system is implementing an anomaly detection model that includes one of: a Gaussian model, a support vector machine model, and a Bayesian network model.

[0021] In some embodiments, the method further comprises: modifying the simulated user bot; and determining an efficacy score of the insider threat detection system based on detections of the simulated user bot and the modified simulated user bot.

[0022] In some embodiments, the first plurality of actions in the computing environment are associated with an administrative user account and the simulated user bot emulates the user behavior of the administrative user account by inserting a plurality of decoy documents into the computing environment.

[0023] In some embodiments, the first plurality of actions in the computing environment are associated with an administrative user account and the simulated user bot emulates the user behavior of the administrative user account by automatically updating a security appliance within the computing environment and determining whether the updated security appliance generates expected state information.

[0024] In some embodiments of the disclosed subject matter, a system for testing insider threat detection systems is provided, the system comprising a hardware processor that is configured to: receive a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts; generate a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts; select a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type; generate a simulated user bot based on the selected model of user behavior; executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment; determine whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and transmit a notification indicating an efficacy of the insider threat detection system based on the determination.

[0025] In accordance with some embodiments of the disclosed subject matter, non-transitory computer-readable media containing computer executable instructions that, when executed by a processor, cause the processor to perform a method for testing insider threat
detection systems is provided. In some embodiments, the method comprises: receiving a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts; generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts; selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type; generating a simulated user bot based on the selected model of user behavior; executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment; determining whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and transmitting a notification indicating an efficacy of the insider threat detection system based on the determination.

[0026] In some embodiments of the disclosed subject matter, a system for testing insider threat detection systems is provided, the system comprising: means for receiving a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts; means for generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts; means for selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type; means for generating a simulated user bot based on the selected model of user behavior; executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment; means for determining whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and means for transmitting a notification indicating an efficacy of the insider threat detection system based on the determination.

**Brief Description of the Drawings**

[0027] FIG. 1 is a diagram of a system suitable for implementing an application that generates and/or injects simulated user bots in accordance with some embodiments of the disclosed subject matter.
FIG. 2 is a diagram of a detection system, such as an insider threat detection system, that is external to a virtual-machine based host, where simulated user bots can be used to test the insider threat detection system, in accordance with some embodiments of the disclosed subject matter.

FIG. 3 is a diagram of a detection system, such as an insider threat detection system, deployed in an enterprise environment with non-virtual machine-based hosts, where simulated user bots can be used to test the insider threat detection system, in accordance with some embodiments of the disclosed subject matter.

FIG. 4 is a diagram of a detection system, such as an insider threat detection system, deployed in a wireless device-based architecture, where simulated user bots can be used to test the insider threat detection system, in accordance with some embodiments of the disclosed subject matter.

FIG. 5 is a diagram of a detection system, such as an insider threat detection system, deployed in a thin client-based architecture, where simulated user bots can be used to test the insider threat detection system, in accordance with some embodiments of the disclosed subject matter.

FIG. 6 is a diagram showing an illustrative example of a process for generating and/or injecting simulated user bots in accordance with some embodiments of the disclosed subject matter.

FIG. 7 is a diagram showing an illustrative example of threats, behaviors, indicators, and detectors in accordance with some embodiments of the disclosed subject matter.

FIG. 8 is a diagram showing an illustrative example of a detection system that implements a Bayesian network, which can be used to model a threat type, behavior, indicators, and detectors, in accordance with some embodiments of the disclosed subject matter.

**Detailed Description**

Generally speaking, the mechanisms described herein relate to generating and injecting simulated user bots into a computing environment. Simulated user bots can, for example, simulate users of different user types (e.g., a particular type of normal user, a particular type of malicious user, etc.). In a more particular example, an agent can be implemented in a computing environment that runs a host application to inject actions of simulated users of one or
more different user types without interfering with the operations of the systems in the computing environment. The injected actions can include the injection of network data and/or host data at various times within the computing environment.

[0036] In some embodiments, the mechanisms can use the simulated user bots to inject actions that emulate a user of a particular user type and can determine whether an insider threat detection system or any other suitable detection system that is monitoring the computing environment detects the actions of the simulated user bot when the simulated user bot is configured to inject actions similar to those of a malicious user.

[0037] It should be noted that the insider threat detection system can implement any suitable anomaly detection model. For example, an insider threat detection system can implement a Gaussian mixture model, a support vector machine, and/or a Bayesian network and the simulated user bots can be used to evaluate one or more of the features of an insider threat detection system in the computing environment.

[0038] In some embodiment, the simulated user bot can be configured and/or modified to test whether the injected actions are detected by the insider threat detection system. For example, a simulated user bot associated with a user account in a computing environment can be changed from a normal user type to a malicious user type (e.g., by modifying simulated user activity to include user actions of a malicious user). In continuing this example, a simulated user bot can inject user actions associated with a normal user type into the computing environment for a first period of time and, for a second period of time, can be modified to inject user actions associated with a malicious user type. The simulated user bot of a testing system can then determine whether a detection system, such as the insider threat detection system, detected the injected user actions associated with the malicious user type that were injected by the modified simulated user bot.

[0039] In a more particular example, the simulated user bots described herein can be configured to generate trace data, which can be used as indicators and detectors in identifying malicious users by insider threat detection systems. These indicators can be temporal statistics derived from an analysis of monitored network logs. These statistics derive group norms from which abnormal users are identified. Controlling the pace and frequency of these trace indicators generated by a simulated user bot can be used to test the insider threat detection system at its margins. This can, for example, provide a detailed analysis of the ease of evading detection by
the insider threat detection system. Moreover, errors in the deployed monitoring infrastructure, either due to bugs in configurations or noise introduced by faulty sensors, can also be revealed if a simulated user bot if undetected (e.g., although directed to purposely exhibit the indicator).

[0040] In another more particular example, the simulated user bots can be used to measure the computational cost of maintaining temporal statistics. For example, low and slow behaviors may cause an insider threat detection system to maintain long term state information for many users, which causes an increasing cost in terms of storage and computation. Accordingly, the use of simulated user bots can provide an approach for evaluating the accuracy of an insider threat detection system and its computational costs.

[0041] It should be noted that, although the embodiments described herein generally relate to testing detection systems, such as an insider threat detection system, and whether such detection systems detect the actions of a simulated user bot, this is merely illustrative.

[0042] In some embodiments, the mechanisms can use the simulated user bots to perform verification tasks within the computing environment. For example, the mechanisms can automatically change the configuration of a security appliance and can use the simulated user bots to inject actions that emulate a user of a particular user type (e.g., by monitoring, recording, modifying, and/or replaying actual user activity) into the computing environment having the newly configured security appliance. In continuing this example, the simulated user bots can test the outcome of such changes to the security appliance by, for example, determining whether the state of the computing environment matches an expected state of the computing environment based on the injected actions. In another example, the simulated user bots can determine whether the newly configured security appliance detected one or more actions injected by the simulated user bots into the computing environment.

[0043] In some embodiments, the simulated user bots can perform security tasks within a computing environment. For example, the mechanisms can use the simulated user bots to inject actions that emulate a user of a particular user type (e.g., an administrative user) into the computing environment, where the injected actions of such an administrative user can place decoy information in one or more particular locations within the computing environment. The use of simulated user bots to place decoy information can, for example, distribute detection capabilities for detecting the exfiltration of data or documents.
[0044] Turning to FIG. 1, an example of a system 100 in which a detection system is implemented and in which a testing mechanism can be implemented to generate and/or inject simulated user bots is shown. As illustrated, system 100 includes multiple collaborating computer systems 102, 104, and 106, a communication network 108, a malicious/compromised computer 110, communication links 112, a detection system 114, and an attacker computer system 116.

[0045] Collaborating systems 102, 104, and 106 can be systems owned, operated, and/or used by universities, businesses, governments, non-profit organizations, families, individuals, and/or any other suitable person and/or entity. Collaborating systems 102, 104, and 106 can include any number of user computers, servers, firewalls, routers, switches, gateways, wireless networks, wired networks, intrusion detection systems, and any other suitable devices. Collaborating systems 102, 104, and 106 can include one or more processors, such as a general-purpose computer, a special-purpose computer, a digital processing device, a server, a workstation, and/or various other suitable devices. Collaborating systems 102, 104, and 106 can run programs, such as operating systems (OS), software applications, a library of functions and/or procedures, background daemon processes, and/or various other suitable programs. In some embodiments, collaborating systems 102, 104, and 106 can support one or more virtual machines. Any number (including only one) of collaborating systems 102, 104, and 106 can be present in system 100, and collaborating systems 102, 104, and 106 can be identical or different.

[0046] It should be noted that the testing mechanisms for generating and/or injected simulated user bots (e.g., for testing detection system 114) can be implemented on one or more of collaborating systems 102, 104, and 106. For example, a collaborating system can deploy a simulated user bot into a computing environment, where the simulated user bot emulates the user actions and behaviors of a malicious user or insider attacker, and can determine whether an insider threat detection system connected to the collaborating system (e.g., over a communication network) detected the simulated user bot. In another example, a collaborating system can deploy a simulated user bot into a computing environment, where the simulated user bot emulates an administrative user by injecting decoy data and/or decoy documents at particular locations within the computer environment to lure covert malware or an insider attacker.

[0047] In a more particular example, a collaborating system implementing the testing mechanisms described herein can access a database, a corresponding translator, and action files.
For example, a collaborating system implementing the testing mechanisms described herein can generate user actions by accessing a database of previously obtained user actions and user behaviors, translating the previously obtained user actions into Python action files. The information contained within the database can include, for example, a unique identifier for a user, a time stamp for a given action, an action column to describe the event that takes place, and a details field that provides additional information on a given action. The details field can contain specific information, such as the exact terms searched for in a search query interface, the title of a page visited, etc. It should be noted that simulated user bots can be generated and injected into a computing environment, such that they are generally identical to users in many facts from having to log in to the speed at which a user edits a document.

[0048] In some embodiments, the testing mechanisms that use simulated user bots can be distributed among multiple collaborating systems. For example, a first collaborating system can monitor and/or provide user actions and, based on the monitored user actions, can generate one or more simulated user bots that emulate the user actions and behaviors of a particular type of user (e.g., an insider attacker), a second collaborating system can inject the simulated user bots into the computing system, and a third collaborating system can communicate and/or monitor an insider threat detection system to determine whether the insider threat detection system detected the simulated user bots. In a more particular example, one of these collaborating systems can calculate performance metrics for the insider threat detection system (e.g., how many simulated user bots were detected, at what computational cost, etc.).

[0049] It should be noted that, in some embodiments, a device implementing the testing mechanisms for generating and/or injecting simulated user bots (e.g., for testing detection systems, such as insider threat detection systems) can be separate from collaborating systems 102, 104, and 106.

[0050] Communication network 108 can be any suitable network for facilitating communication among computers, servers, etc. For example, communication network 108 can include private computer networks, public computer networks (such as the Internet), telephone communication systems, cable television systems, satellite communication systems, wireless communication systems, any other suitable networks or systems, and/or any combination of such networks and/or systems. For example, simulated user bots can inject user actions over communication network 108.
Malicious/compromised computer 110 can be any computer, server, or other suitable device that includes the covert malware. In addition, malicious/compromised computer 110 can be used to launch a computer threat, such as a virus, worm, trojan, rootkit, spyware, key recovery attack, denial-of-service attack, malware, probe, etc. The owner of malicious/compromised computer 110 can be any university, business, government, non-profit organization, family, individual, and/or any other suitable person and/or entity.

It should be noted that, in some embodiments, an external attacker can become an inside attacker when the external attacker attains internal network access. For example, using spyware, rootkits, or any other suitable malware, external attackers can gain access to communications network 108. Such software can easily be installed on computer systems from physical or digital media (e.g., email, downloads, etc.) that provides an external attacker with administrator or "root" access on a machine along with the capability of gathering sensitive data. The external attacker can also snoop or eavesdrop on one or more systems 102, 104, and 106 or communications network 108, download and exfiltrate data, steal assets and information, destroy critical assets and information, and/or modify information. Rootkits have the ability to conceal themselves and elude detection, especially when the rootkit is previously unknown, as is the case with zero-day attacks. An external attacker that manages to install rootkits internally in effect becomes an insider, thereby multiplying the ability to inflict harm.

In some embodiments, the owner of malicious/compromised computer 110 may not be aware of what operations malicious/compromised computer 110 is performing or may not be in control of malicious/compromised computer 110. Malicious/compromised computer 110 can be acting under the control of another computer (e.g., attacker computer system 116) or autonomously based upon a previous computer attack which infected computer 110 with a virus, worm, trojan, spyware, malware, probe, etc. For example, some malware can passively collect information that passes through malicious/compromised computer 110. In another example, some malware can take advantage of trusted relationships between malicious/compromised computer 110 and other systems 102, 104, and 106 to expand network access by infecting other systems. In yet another example, some malware can communicate with attacking computer system 116 through an exfiltration channel 120 to transmit confidential information (e.g., IP addresses, passwords, credit card numbers, etc.).
[0054] It should be noted that any number of malicious-compromised computers 110 and attacking computer systems 116 can be present in system 100, but only one is shown in FIG. 1 to avoid overcomplicating the drawing. It should also be noted that malicious-compromised computer 110 and/or attacking computer system 116 may have been present in system 100 at a particular point in time in which user action information, such as network data and host data, can be monitored and/or captured for use in generating simulated user bots that emulate such malicious users.

[0055] More particularly, for example, each of the one or more collaborating or client computers 102, 104, and 106, malicious-compromised computer 110, detection system 114, and attacking computer system 116, can be any of a general purpose device such as a computer or a special purpose device such as a client, a server, etc. Any of these general or special purpose devices can include any suitable components such as a processor (which can be a microprocessor, digital signal processor, a controller, etc.), memory, communication interfaces, display controllers, input devices, etc. For example, client computer 1010 can be implemented as a personal computer, a personal data assistant (PDA), a portable email device, a multimedia terminal, a mobile telephone, a set-top box, a television, etc.

[0056] In some embodiments, any suitable computer readable media can be used for storing instructions for performing the processes described herein, can be used as a content distribution that stores content and a payload, etc. For example, in some embodiments, computer readable media can be transitory or non-transitory. For example, non-transitory computer readable media can include media such as magnetic media (such as hard disks, floppy disks, etc.), optical media (such as compact discs, digital video discs, Blu-ray discs, etc.), semiconductor media (such as flash memory, electrically programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), etc.), any suitable media that is not fleeting or devoid of any semblance of permanence during transmission, and/or any suitable tangible media. As another example, transitory computer readable media can include signals on networks, in wires, conductors, optical fibers, circuits, any suitable media that is fleeting and devoid of any semblance of permanence during transmission, and/or any suitable intangible media.

[0057] Referring back to FIG. 1, communication links 112 can be any suitable mechanism for connecting collaborating systems 102, 104, 106, malicious/compromised
computer 110, detection system 114, and attacking computer system 116 to communication
network 108. Links 112 can be any suitable wired or wireless communication link, such as a T1
or T3 connection, a cable modem connection, a digital subscriber line connection, a Wi-Fi or
802.11(a), (b), (g), or (n) connection, a dial-up connection, and/or any other suitable
communication link. Alternatively, communication links 112 can be omitted from system 100
when appropriate, in which case systems 102, 104, and/or 106, computer 110, and/or detection
system 114 can be connected directly to communication network 108.

[0058] Detection system 114 can be any computer, server, router, or other suitable device
for modeling, generating, inserting, distributing, monitoring, verifying, and/or managing decoy
information into system 100. Similar to collaborating systems 102, 104, and 106, detection
system 114 can run programs, such as operating systems (OS), software applications, a library of
functions and/or procedures, background daemon processes, and/or various other suitable
programs. In some embodiments, detection system 114 can support one or more virtual
machines.

[0059] In a more particular example, detection system 114 can be implemented in a
virtual machine environment, where an out-of-host agent drives simulated user activity that is
meant to convince covert malware residing within the guest operating system that it has captured
legitimate credentials. This is generally applicable to systems that are fully virtualized (e.g.,
VMWare) and the operating systems on which they are supported. An illustrative example of
detection system 114 implemented in a virtual machine architecture is shown in FIG. 2.

[0060] As shown, architecture 200 can include a simulation and injection component 210
(sometimes referred to herein as "VMSim" or a "simulation engine"), a virtual machine
verification (VMV) component 220, and a network monitoring component 230. Simulation and
injection component 210 executes outside of a virtual machine and passes its actions (e.g., user
actions 240 and simulated user activity or decoys 250) into a guest operating system 260. More
particularly, simulation and injection component 210 generates simulated user activity 250 by
recording, modifying, and replaying keyboard and mouse events captured from users. In
addition, simulation and injection component 210 can replay and inject monitored user activity
(without decoys) to increase the believability of the simulated user activity 250. Upon the
injection of simulated user activity 250, virtual machine verification component 220 can be used
to determine whether the state of the virtual machine is an expected state (e.g., one of a number
of predefined states). Network monitoring component 230 can then detect when covert malware or a simulated user bot attempts to exfiltrate data. For example, network monitoring component 230 records and transmits alerts in response to determine that malicious traffic is originating from the virtual machine host. In some embodiments, verification component 220 or any other suitable component in FIG. 2 can be used to determine whether the simulated user activity 250 corresponding to a malicious user or an insider attacker was detected by an insider threat detection system.

[0061] Alternatively or additionally, detection system 114 and its detection mechanisms can be deployed in an enterprise environment. For example, detection system 114 can be used in an enterprise environment to monitor for site-specific credential misuse and to profile attackers targeting that environment. In a more particular example, detection system 114 can be deployed to run simulations on a user's system (e.g., one of collaborating systems 102, 104, or 106) when it is idle (e.g., during meetings, at particular times during the night, etc.). Virtual machines can be created on demand from a user's native environment. For example, as shown in FIG. 3, detection system 114 is deployed as an enterprise service that runs a simulation over exported copies of multiple users' disk images 310 from corresponding user computers 320. Alternatively, in some embodiments, the machine state of each user computer 320 can be synchronized with the state of the detection system 114. As a result, detection system 114 including, for example, a simulation and injection component 210 of FIG. 2, can use the disk images 310 to simulate user activity and inject the simulated user activity into the enterprise environment 300. This allows detection system 114 to detect covert malware conducting long-term corporate reconnaissance. For example, detection system 114 can be used to detect covert malware that attempts to steal credentials only after they have been repeatedly used in the past. That is, instead of generally detecting covert malware, detection system 114 can be used to detect targeted espionage software.

[0062] In some embodiments, although the embodiment described above relates to detection system 114 simulating user activity and injecting the simulated user activity to catch covert malware, user computers 320 or any other suitable computing device can generate simulated user activity and deploy a simulated user bot into the enterprise environment 300, where the simulated user bot emulates the user actions and behaviors of a malicious user or insider attacker. In continuing this example, user computers 320 or any other suitable computing
device implementing the testing mechanisms described herein can then determine whether an insider threat detection system, such as detection system 114, detected the simulated user bot. In another example, a first user computer 320 can monitor and/or provide user actions and, based on the monitored user actions, can generate one or more simulated user bots that emulate the user actions and behaviors of a particular type of user (e.g., an insider attacker). A second user computer 320 can inject the simulated user bots into the enterprise environment 300, and a third user computer 320 can communicate and/or monitor detection system 114 to determine whether detection system 114 detected the simulated user bots. In a more particular example, one of the user computers 320 can calculate performance metrics for detection system 114 (e.g., how many simulated user bots were detected, how long did it take to detect a particular simulated user bot from the time it was injected into the enterprise environment, at what computational cost, etc.).

It should be noted that, in some embodiments, specialized decoy information and general decoy information can be generated, where specialized decoys are used to detect targeted espionage software and where general decoys can be used to assist the organization identify compromised internal users. In some embodiments, simulated user bots can emulate an administrative user by generating and/or placing decoy information within the computing environment to, for example, lure internal users to exfiltrate the decoy information or detect whether detection system 114 detects the exfiltration of the decoy information placed by a simulated user bot.

In some embodiments, a testing system 405 and its testing mechanisms can be implemented without using virtual machines. For example, a wireless device-based architecture 400, as shown in FIG. 4, provides a testing device 405 that injects a simulated user bot (e.g., having mouse and keyboard events based on user activity information) wirelessly using the Bluetooth protocol or any other suitable wireless protocol into user computers 410 via wireless communication paths 420. In a more particular example, testing device 405 can run a Bluetooth proxy application that receives user activity (e.g., by monitoring network traffic), translates the user activity to Bluetooth human interface device (HID) protocol, and transmits them to a host, such as one of user computers 410. Testing device 405 can then verify the success and failure of an insider threat detection system monitoring user computers 410 using traffic analysis of encrypted protocols. For example, as shown in FIG. 4, network traffic 430 can be monitored and portions of the network traffic can be verified to determine whether an insider
threat detection system, which is monitoring user computers 410, detected the simulated user bot injected into the computing environment and/or inhibited the simulated user bot from proceeding. In another example, testing device 405 can run a Bluetooth proxy application that monitors user activity of an administrative user and generates a simulated user bot that emulates the user actions of the administrative user, which, in turn, places decoy information within the computing environment to distribute monitoring capabilities for the detection of insider attackers attempting to exfiltrate data and/or documents.

[0065] In yet another suitable embodiment where detection system 114 and its detection mechanisms can be implemented without using virtual machines, FIG. 5 shows a thin-client based architecture 500 having detection system 114 implemented as a thin client. As shown, thin client-based architecture generally includes a central virtual machine host 510 (which can be one physical server or multiple servers) and one or more dummy computers 520 connected to the host via communication paths 530 (e.g., a local and fast network connection). Detection system 114 and other thin clients 520, which can include a testing system, transmit user actions (e.g., keyboard events, mouse events, etc.) to central virtual machine host 510 and remotely display the screen output of the virtual machine. That is, particular computations and functionality can be offloaded to host 510. Using thin clients 520, each user can access and use virtual machines hosted on central virtual machine host 510 and detection system 114 can access each hosted virtual machine.

[0066] More particularly, detection system 114 is deployed as a thin client (outside of the virtual machines) that periodically connects to each hosted virtual machine and detects whether there is covert malware on one or more of the thin clients. In order to test detection system 114, a testing system can inject a simulated user bot within the computing environment. The remote access protocols used in thin client environments (e.g., Citrix, VNC, remote desktop protocol (RDP), etc.) can be used for both injecting simulated user activity or any other suitable decoy information and verification. For example, a testing system in the thin client environment can inject a simulated user bot that employs decoy credentials into a hosted virtual machine and can then perform a verification of the injected simulated user bot. In continuing this example, detection system 114 can receive arbitrary portions of rendered screens and count the absolute number of pixels in each of the portions and the testing system can determine whether detection system 114 detects the simulated user bot and its actions.
[0067] It should be noted that detection system 114 and/or a testing system can generate
decoy information (e.g., bogus credentials) that complies with particular properties that enhance
the deception for different classes or threat levels of inside attackers. Decoy information can be
generated that is, for example, believable, enticing, conspicuous, detectable, variable,
differentiable from actual or authentic information, non-interfering with legitimate users, etc.

[0068] Detection system 114 and/or a testing system can generate decoy information that
is believable. That is, decoy information can be generated such that it is difficult for a user to
discern whether the decoy information is from a legitimate source or in fact a decoy. For
example, decoy information can be generated to appear realistic and indistinguishable from
actual information used in the system. More particularly, detection system 114 and/or a testing
system can record information, events, and network flow in systems 100, 200, 300, 400, and 500.
For example, detection system 114 and/or a testing system can record user activity, such as
keyboard and mouse events, modify the recorded user activity to simulate believable decoy
information in the form of simulated user activity. In addition, detection system 114 and/or a
testing system can replay recorded user activity captured from real users that is not used to
simulate user activity, but is used to support the believability of simulated user activity.
Accordingly, using actual user activity, simulated user activity, and/or a model of user activity as
described herein, covert malware or any other suitable attacking computer does not detect
detection system 114 and/or a testing system as the source of decoy information.

[0069] In some embodiments, detection system 114 and/or a testing system can
determine whether decoy information complies with a believability property. For example,
detection system 114 and/or a testing system can perform a decoy Turing test, where portions of
decoy information and legitimate information are selected – one contains decoy information and
the other contains information randomly selected from authentic information. The two pieces of
information can be presented to a volunteer or any other suitable user and the volunteer can be
tasked to determine which of the two are authentic. In some embodiments, in response to testing
the believability of decoy information and receiving a particular response rate, detection system
114 and/or a testing system can consider decoy information to comply with the believability
property. For example, detection system 114 and/or a testing system can determine whether a
particular piece of decoy information, such as a bogus credential, is selected as an authentic and
believable piece of information at least 50% of the time, which is the probability if the volunteer
user selected at random. In another example, detection system 114 and/or a testing system can allow a user, such as an administrator user that has access to detection system 114 and/or a testing system, to select a particular response rate for the particular type of decoy information. If the decoy information is tested for compliance with the believability property and receives an outcome less than the predefined response rate, detection system 114 and/or a testing system can discard the decoy information and not inject the decoy information in the computing environment.

[0070] Similarly, detection system 114 and/or a testing system can also determine whether simulated user activity complies with a believability property. For example, detection system 114 and/or a testing system can perform a Turing test, where portions of simulated user activity and actual user activity are selected. The two pieces of information can be presented to a volunteer or any other suitable user and the volunteer can be tasked to determine which of the two are authentic.

[0071] Accordingly, decoy information that complies with one or more of the above-mentioned properties can be used to entice covert malware into believing that it has obtained legitimate credentials and confuse or slow down covert malware. For example, covert malware can be forced to spend time and energy obtaining information and then sorting through the collected information to determine actual information from decoy information. In another example, the decoy information can be modeled to contradict the actual or authentic data on one of systems 100, 200, 300, 400, or 500, thereby confusing the covert malware at attacking computer system 116 or the user of attacking computer system 116 and luring the user of attacking computer system 116 to risk further actions to clear the confusion.

[0072] As used herein, decoy information can be generated by the testing system or any other suitable system and placed by a simulated user bot generated by the testing system within the computing environment to detect the attempted exfiltration of data or documents from the computing environment. For example, by using one or more simulated user bots to place decoy information in multiple locations within the computing environment, the simulated user bots can be used to distribute security and monitor capabilities.

[0073] As described above, mechanisms for generating and injecting simulated user bots into a computing environment are provided. FIG. 6 illustrates an example of a process 600 for generating and injecting simulated user bots into a computing environment in accordance with
some embodiment of the disclosed subject matter. As shown, process 600 begins by monitoring user activity at 602. The user activity can include, for example, mouse and keyboard events captured from users (e.g., users at collaborating system 102 of FIG. 1), network traffic, etc. For example, as shown in FIG. 2, simulation and injection component 210 can receive recorded mouse and keyboard events (e.g., X-Window events) captured from users. In another example, as shown in FIG. 4, a testing system can monitor traffic and conversation summaries to determine user activity over a network. In yet another example, as shown in FIG. 5, a testing system can receive monitored mouse and keyboard actions from users on user computers 520.

[0074] In a more particular example, process 600 can obtain user actions by accessing a database of previously obtained user actions and user behaviors. The information contained within the database can include, for example, a unique identifier for a user, a time stamp for a given action, an action column to describe the event that takes place, and a details field that provides additional information on a given action. The details field can contain specific information, such as the exact terms searched for in a search query interface, the title of a page visited, etc.

[0075] In some embodiments, simulated user activity can then be generated based on the monitored user activity at 604. For example, as shown in FIG. 2, simulation and injection component (VMSim) 210 can perform a simulation process that records, modifies, and replays mouse and keyboard events based on the monitored user activity.

[0076] In continuing this example, process 600 can use a simulated user bot database translator that converts the fields in the database into an action file. For example, the simulated user bot database translator can receive a database stored in SQL with at least the fields user ID, event, and timestamp, and convert the fields into Python action files for users.

[0077] In some embodiments, one or more rules can be applied to the simulated user bot database translator. For example, the simulated user bot database translator can derive the browser application used by a particular user by searching through the information provided in a given browser tab if the information is not provided otherwise. In another example, the simulated user bot database translator can perform the search with the same service as a given user by searching through the information contained in the browser tab unless the information is provided in another field. In yet another example, the simulated user bot database translator can search through the browser tab for information regarding the email service used unless provided
explicitly by the dataset in the database. In a further example, the simulated user bot database translator can search a social media site specified in the browser tab unless otherwise provided in the dataset in the database. In another further example, the simulated user bot database translator can browse to the website referred to if the browser tab contains a full HTTP URL. It should be noted that any suitable number and any suitable combination of rules can be applied to the simulated user bot database translator to create action files.

[0078] In continuing this example, upon translating the database into action files for each of the users (or user accounts), process 600 can select one or more of these action files for use in generating simulated user bots. For example, process 600 can analyze the records associated with each user in the action files and determine to remove particular users. By removing particular users and their associated action files, process 600 can obtain a set of baseline actions for users.

[0079] In some embodiments, process 600 can generate a modified action file by adding noise or changing normal behavior of an action file such that a user appears malicious. These modified action files can, for example, be used to generate a simulated user bot.

[0080] Alternatively, in systems in which malicious/compromised computers 110 and/or attacking computer systems 116 are present and detected by detection system 114, process 600 can analyze the user activity information associated with these computing systems and generate one or more simulated user bots that emulate a malicious user. It should be noted that each simulated user bot can include a different set and/or varying levels of malicious user activities. As such, process 600 can inject multiple simulated user bots that mimic different types of malicious users in order to determine whether a particular type of malicious user or a particular action by a malicious user may evade detection by detection system 114.

[0081] Alternatively, in some embodiments, process 600 can derive general parameters for the users from the action files and use the derived general parameters to provide actions for the generation of a simulated user bot. For example, for a social media site visit, process 600 can receive a specified list of the top ten social media sites and can arbitrarily or randomly select one social media site for a particular user and then calculate how long a user should spend on that given site based on the history of how long other users spent on that particular site in the past. This random selection process that is used to provide actions for generating simulated user bots can cause a simulated user to be unique from another simulated user.
Referring back to FIG. 6, process 600 can generate one or more simulated user bots using the action files at 606. Such a simulated user bot can perform a task that a regular user would perform, such as logging into an account, sending an email, opening a website, creating and modifying a document, etc. In the case of an administrative user, a simulated user bot can verify the detection mechanisms of an insider threat detection system, place decoy documents within a computing environment, update security appliances and determine whether the recently updated security appliance is provided expected results, etc.

In a more particular example, an example of a script and action in a formal language that specifies a sequence of user activity is shown in FIG. 7. As shown in FIG. 7, particular actions and/or modified actions of an actual user can be mapped to the constructs of the formal language. It should be noted that the formal language shown in FIG. 7 can be used to differentiate between different types of user activity. For example, in analyzing the timestamps of the monitored user activity, process 600 can determine a gap between activities. It should be noted that process 600 can determine that it is unrealistic to assume that a user proceeds directly from one task to the next without a pause. The delta in time between activities in the database can allow process 600 to train simulated user bots to have pauses between actions, thereby improving the overall realism of a simulated user bot. In some embodiments, one or more checks can be provided to determine whether a prescribed action has completed execution and that the computing system is in an expected state prior to continuing to the next action. If such a check fails, process 600 can attempt to execute the action again prior to continuing on to the subsequent actions.

It should be noted that simulated user bots can be generated and injected into a computing environment, such that they are generally identical to users in many facts from having to login to the speed at which a user edits a document.

In some embodiments, once the simulated user activity is generated, one or more models can be applied. For example, the simulated user activity can be tuned to one or more biometric models for keystroke speed, mouse speed, mouse distance, and the frequency of errors made by a user when typing. These parameters function as controls over the formal language and assist in creating variability in the simulations. Depending on the particular simulation, other parameters such as uniform resource locators (URLs) or other text that must be typed are then entered to adapt each action.
[0086] In some embodiments, one or more features, such as keycodes (e.g., the ASCII code representing a key), the duration for which a key is pressed, keystroke error rates, mouse movement speed, and mouse movement distance, can be recorded for the construction of one or more user models or biometric models. For example, generative models for keystroke timing can be created by dividing the recorded data for each keycode pair into separate classes, where each class is determined by the distance in standard deviations from the mean. The distribution for each keycode sequence can be calculated as the number of instances of each class. Simulation keystroke timing can be adapted to profiles of individual users by generating random times that are bounded by the class distribution.

[0087] It should be noted that, in some embodiments, alternative applications can be executed by the simulated user bot in place of a software application executed by a monitored user. For example, in response to detecting that an action file of a user includes the execution of a commercial software application (e.g., Microsoft Office), process 600 can replace the execution of the commercial software application with an open source equivalent that corresponds to the commercial software application (e.g., Apache OpenOffice). This can, for example, not compromise the realism of the simulated user bot while avoiding potential software licensing issues.

[0088] Referring back to FIG. 6, the one or more simulated user bots can be injected into a computing environment at 608.

[0089] As shown in FIGS. 2-5, the simulated user bots can be injected using any suitable number of approaches. Referring back to FIG. 2, simulation and injection component (VMSim) 210 transmits the simulated user bots into the guest operating system 260. In a more particular example, simulation and injection component 210 obtains access to the display of guest operating system 260 to play back the simulated user activity of a simulated user bot. During playback, simulation and injection component 210 can automatically detect the position of the virtual machine window and adjust the coordinates to reflect the changes of the simulated user activity. Alternatively, a simulated user bot can use simulation and injection component 210 to transmit decoy information into a suitable buffer.

[0090] In some embodiments, as shown in FIG. 4, the simulated user bot can be injected by simulating Bluetooth input devices. In selecting the Bluetooth protocol, the physical proximity of the workstations to one another within a typical workspace can be leveraged. More
particularly, a Bluetooth proxy application can be used to transmit the simulated user activity. The Bluetooth proxy application can receive user activity from GNU Xnee or any other suitable function, modify and translate the user actions to the Bluetooth human interface devices (HID) protocol, and transmit the simulated user activity into a host.

[0091] Alternatively or additionally, in the thin client environment of FIG. 5, remote access protocols (e.g., Citrix, VNC, etc.) can be used to inject the simulated user activity of a simulated user bot. A testing system can be deployed as a thin client that periodically connects to each hosted virtual machine and injects simulated user bots.

[0092] It should be noted that simulated user bots that emulate the behavior of normal users can be injected along with simulated user bots that emulate the behavior of malicious users. This can, for example, support the believability of the simulated user activity. This can also, for example, be used to determine whether a detection system detects the simulated user activity performed by a simulated user bot that is emulating the behavior of a malicious user as opposed to the simulated user activity performed by a simulated user bot that is emulating the behavior of a normal user.

[0093] In some embodiments, the simulated user bot that has been injected into the computing environment can be modified from emulating the behavior of a normal user having a normal user type to the behavior of a malicious user having an abnormal user type. This can, for example, be used to evaluate whether the detection system detects the simulated user activity of the simulated user bot in response to the modified behavior and, if it does detect the modified user activity, it can be determined how long it takes and the computational resources expended to make such a detection.

[0094] In some embodiments, in response to injecting one or more simulated user bots into the computing environment, process 600 can record the logs at the end of a run by a simulated user bot. It should be noted that, in some embodiments, a virtual connection is used to ensure that there are no extraneous activities conducted locally and recorded in such logs.

[0095] After the data has been generated from a simulated user bot, process 600 can analyze the recorded data that would trigger user anomaly detectors at 610, such as an insider threat detection system. An illustrative example of threat types, behaviors, indicators, and detectors are shown, for example, in FIG. 8.
It should be noted that the user anomaly detectors for detecting anomalous behavior can include, for example, a Gaussian mixture model, a support vector machine, and/or a Bayesian network.

For example, a Gaussian mixture model used in user anomaly detectors is a probabilistic model, which can be represented by:

$$
p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i),
$$

where x is a D-dimensional continuous value data vector, w_i are mixture weights and g() are component Gaussian densities. It should be noted that these Gaussian Mixture Models include the covariance structure of the data.

In another example, support vector machines used in user anomaly detectors can generally include supervised classifiers that attempt to map input vectors into a high dimensional feature space using optimal hyperplanes, those with the maximal margin between the vectors of the two classes, for separable classes. Support vector machines can attempt to find a function that is positive on a subset of the input space and negative on the complement by mapping the input data into a higher dimensional space and using the origin as a negative training point. The objective function can be represented by:

$$
\min_{w \in F, \xi \in R^l, \rho \in R} \frac{1}{2}||w||^2 + \frac{1}{vl} \sum_i \xi_i - \rho
$$

subject to \((w \cdot \Phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0\)

where \(v\) can be a parameter between 0 and 1 that controls how tightly the support vector machine fits the data.

In yet another example, Bayesian networks used in user anomaly detectors can make use of probabilistic relationships among variables of interest in an acyclic graphical model. For example, FIG. 9 shows an illustrative example of how such a Bayesian network can be used
to model a threat type, behavior, indicators, and detectors. A Bayesian network can be represented as follows:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \pi_i)$$

where $X_1, X_2, \ldots, X_n$ represent random variables and $\pi_i$ is the set of parents of $X_i$.

[0100] In some embodiments, the simulated user bots described herein can be configured to generate trace data, which can be used as indicators and detectors in identifying malicious users by insider threat detection systems. These indicators can be temporal statistics derived from an analysis of monitored network logs. These statistics derive group norms from which abnormal users are identified. Controlling the pace and frequency of these trace indicators generated by a simulated user bot can be used to test the insider threat detection system at its margins. This can, for example, provide a detailed analysis of the ease of evading detection by the insider threat detection system. Moreover, errors in the deployed monitoring infrastructure, either due to bugs in configurations or noise introduced by faulty sensors, can also be revealed if a simulated user bot if undetected (e.g., although directed to purposely exhibit the indicator).

[0101] In a more particular example, process 600 can use the challenge problems from an external third party operating and analyzing a deployed insider threat solution as a guide to set a threat type, behavior, indicators, and detectors. For example, the threat type can be defined as individuals using their computing devices with abnormal work habits, where the behavior associated with this threat is a user who uses their computing device outside of normal work hours (e.g., between 5:00:01 PM and 6:59:59 AM EST). Three illustrative indicators of this behavior can be: 1) in the top five percent of the daily frequency average distribution of Google or Bing searches conducted between 5:00:01 PM and 6:59:59 AM EST; 2) in the top five percent of the daily frequency average distribution of social media website visits between 5:00:01 PM and 6:59:59 AM EST; and 3) in the top five percent of the daily frequency average distribution of actions on files and documents between 5:00:01 PM and 6:59:59 AM EST. This, in turn, can correspond to the following three detectors: 1) at least 13 log entries for a Google or Bing search conducted between 5:00:01 PM and 6:59:59 AM EST; 2) at least 61 log entries for a social media website visit between 5:00:01 PM and 6:59:59 AM EST; and 3) at least 90 log entries for actions on files and documents between 5:00:01 PM and 6:59:59 AM EST.
In continuing this example, anomalous users can be selected from the dataset and, in some cases, a subset of the anomalous users can be selected (e.g., by random selection). The user activities associated with the selected users can, for example, be supplemented with behaviors to appear anomalous. A simulated user bot based on the modified actions of an anomalous user can be executed within a computing environment in which a detection system is monitoring the computing environment (e.g., one of the Gaussian mixture model, the support vector machine, the Bayesian network). In response, process 600 can determine whether each of the detection systems detected the injected malicious or anomalous user bot.

It should be noted that simulated user bots having differing levels of maliciousness can be generated to continue to test these detection systems. For example, in response to determining that a malicious user bot has not been detected by a particular detection system, process 600 can transmit a notification that indicates parameters for improving the performance of the detection system at 612. In another example, in response to determining that a malicious user bot has been detected by a particular detection system, process 600 can continue to modify the simulated user actions until the detection system fails to detect the simulated user actions as being anomalous. In this example, the testing mechanisms described herein can determine the bounds of a detection system and can, for example, transmit a notification that indicates parameters that may be used to evade detection, thereby allowing an administrative user to modify the detection system such that detection is not evaded. In a more particular example, the notification can include a modification to an indicator or rule applied by the detection system such that detection of a simulated user bot would not be evaded.

This can, for example, provide a detailed analysis of the ease of evading detection by the insider threat detection system. Moreover, errors in the deployed monitoring infrastructure, either due to bugs in configurations or noise introduced by faulty sensors, can also be revealed if a simulated user bot if undetected (e.g., although directed to purposely exhibit the indicator).

In some embodiments, the simulated user bots can be used to measure the computational cost of maintaining temporal statistics, where the measurements can be included in the notification at 612. For example, low and slow behaviors may cause an insider threat detection system to maintain long term state information for many users, which causes an increasing cost in terms of storage and computation. Accordingly, the use of simulated user bots
can provide an approach for evaluating the accuracy of an insider threat detection system and its computational costs.

[0106] It should be noted that, although the embodiments described herein generally relate to testing detection systems, such as an insider threat detection system, and whether such detection systems detect the actions of a simulated user bot that is emulating a malicious user or an insider attacker, this is merely illustrative.

[0107] In some embodiments, the simulated user bots can be injected into a computing environment to perform verification tasks within the computing environment. For example, a simulated user bot can monitor the user actions of an administrative user (e.g., or any suitable user having a particular level of authorizations or permissions within the computing environment) and can perform particular verification tasks, such as automatically updating the configuration of a security appliance and determine whether the newly configured security appliance provides an expected state. In continuing this example, the simulated user bots can test the outcome of such changes to the security appliance by, for example, determining whether the state of the computing environment matches an expected state of the computing environment based on the injected actions. In another example, the simulated user bots can determine whether the newly configured security appliance detected one or more actions injected by the simulated user bots into the computing environment.

[0108] For example, process 600 can continue by performing a verification that includes determining whether state information of the application or security appliance matches an expected state after the simulated user activity is injected. Process 600 verifies the success or failure of user events that are passed to, for example, a guest operating system. For example, in some embodiments, a visual verification can be performed by determining whether the screen output changed in response to simulated user activity (e.g., with respect graphical artifacts or pixel selections).

[0109] In some embodiments, the simulated user bots can perform security tasks within a computing environment. For example, the mechanisms can use the simulated user bots to inject actions that emulate a user of a particular user type (e.g., an administrative user) into the computing environment, where the injected actions of such an administrative user can place decoy information in one or more particular locations within the computing environment. The
use of simulated user bots to place decoy information can, for example, distribute detection capabilities for detecting the exfiltration of data or documents.

[0110] Accordingly, methods, systems, and media for testing insider threat detection systems are provided.

[0111] Any suitable hardware and/or software can be used to perform the mechanisms described herein. For example, a general purpose device such as a computer or a special purpose device such as a client, a server, etc. can be used to execute software for performing the mechanisms described herein. Any of these general or special purpose devices can include any suitable components such as a hardware processor (which can be a microprocessor, digital signal processor, a controller, etc.), memory, communication interfaces, display controllers, input devices, etc. This hardware and/or software can be implemented as part of other equipment or can be implemented as stand-alone equipment (which can be coupled to other equipment).

[0112] In some embodiments, any suitable computer readable media can be used for storing instructions for performing the functions and/or processes herein. For example, in some embodiments, computer readable media can be transitory or non-transitory. For example, non-transitory computer readable media can include media such as non-transitory forms of magnetic media (such as hard disks, floppy disks, and/or any other suitable magnetic media), non-transitory forms of optical media (such as compact discs, digital video discs, Blu-ray discs, and/or any other suitable optical media), non-transitory forms of semiconductor media (such as flash memory, electrically programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), and/or any other suitable semiconductor media), any suitable media that is not fleeting or devoid of any semblance of permanence during transmission, and/or any suitable tangible media. As another example, transitory computer readable media can include signals on networks, in wires, conductors, optical fibers, circuits, any suitable media that is fleeting and devoid of any semblance of permanence during transmission, and/or any suitable intangible media.

[0113] Although the invention has been described and illustrated in the foregoing illustrative embodiments, it is understood that the present disclosure has been made only by way of example, and that numerous changes in the details of implementation of the invention can be made without departing from the spirit and scope of the invention. Features of the disclosed embodiments can be combined and rearranged in various ways.
What is claimed is:

1. A method for testing insider threat detection systems, the method comprising:
   receiving, using a hardware processor, a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts;
   generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts;
   selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type;
   generating a simulated user bot based on the selected model of user behavior;
   executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment;
   determining whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and
   transmitting a notification indicating an efficacy of the insider threat detection system based on the determination.

2. The method of claim 1, wherein each of the first plurality of actions includes a user identifier associated with one of the plurality of user accounts, an event, and a timestamp for the event.

3. The method of claim 1, wherein each of the first plurality of actions includes a description of the event.

4. The method of claim 1, wherein the description of the event includes at least one of search terms and page visitation information.

5. The method of claim 1, wherein at least one of the first plurality of actions includes a user selection of one or more particular documents in the computing environment.
6. The method of claim 1, further comprising classifying each of the first plurality of user actions into a category of a plurality of categories of user actions.

7. The method of claim 1, further comprising:
   storing the first plurality of actions in a storage device;
   translating the first plurality of actions in the storage device by applying a plurality of rules; and
   generating an action file for each of the plurality of user accounts based on the translated plurality of actions.

8. The method of claim 7, further comprising modifying the action file by removing at least one of the first plurality of actions.

9. The method of claim 1, further comprising:
   deriving user parameters based on the translated plurality of actions; and
   generating an action file based on the derived user parameters.

10. The method of claim 10, wherein the user parameters are derived from a subset of actions that are randomly selected from the translated plurality of actions.

11. The method of claim 1, further comprising:
    creating a user account in the computing environment; and
    introducing the simulated user bot into the computing environment using the created user account.

12. The method of claim 1, further comprising analyzing the second plurality of actions by determining whether the insider threat detection system identifies the simulated user bot as the malicious user.
13. The method of claim 1, wherein the insider threat detection system is implementing an anomaly detection model that includes one of: a Gaussian model, a support vector machine model, and a Bayesian network model.

14. The method of claim 1, further comprising:
   modifying the simulated user bot; and
   determining an efficacy score of the insider threat detection system based on detections of the simulated user bot and the modified simulated user bot.

15. The method of claim 1, wherein the first plurality of actions in the computing environment are associated with an administrative user account and wherein the simulated user bot emulates the user behavior of the administrative user account by inserting a plurality of decoy documents into the computing environment.

16. The method of claim 1, wherein the first plurality of actions in the computing environment are associated with an administrative user account and wherein the simulated user bot emulates the user behavior of the administrative user account by automatically updating a security appliance within the computing environment and determining whether the updated security appliance generates expected state information.

17. A system for testing insider threat detection systems, the system comprising:
   a hardware processor that is configured to:
   receive a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts;
   generate a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts;
   select a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type;
   generate a simulated user bot based on the selected model of user behavior;
execute the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment; determine whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and transmit a notification indicating an efficacy of the insider threat detection system based on the determination.

18. A non-transitory computer-readable medium containing computer-executable instructions that, when executed by a processor, cause the process to perform a method for testing insider threat detection systems, the method comprising:

- receiving a first plurality of actions in a computing environment that are associated with one of a plurality of user accounts;
- generating a plurality of models of user behavior based at least in part on the first plurality of actions, wherein each of the plurality of models of user behavior is associated with each of the plurality of user accounts;
- selecting a model of user behavior from the plurality of models of user behavior, wherein the model of user behavior is associated with a malicious user type;
- generating a simulated user bot based on the selected model of user behavior;
- executing the simulated user bot in the computing environment, wherein the simulated user bot injects a second plurality of actions in the computing environment;
- determining whether an insider threat detection system executing within the computing environment identifies the simulated user bot as a malicious user; and
- transmitting a notification indicating an efficacy of the insider threat detection system based on the determination.
FIG. 1
FIG. 2

200

USER ACTIONS 240

DECLOYS 250

SIMULATION AND INJECTION COMPONENT (VMSim)

GUEST OPERATING SYSTEM

VIRTUALIZATION LAYER

VERIFICATION COMPONENT (VMV)

NETWORK MONITOR

HOST OPERATING SYSTEM

VERIFICATION 220
FIG. 3
START

MONITOR USER ACTIVITY (E.G., MOUSE AND KEYBOARD EVENTS, NETWORK TRAFFIC, ETC. ASSOCIATED WITH ONE OR MORE USER ACCOUNTS)

GENERATE A PLURALITY OF MODELS OF USER BEHAVIOR BASED ON THE USER ACTIVITY INFORMATION (E.G., TRANSLATING A DATABASE INTO ACTION FILES AND SELECTING AND/OR MODIFYING ACTION FILES FOR SIMULATED USER BOTS)

GENERATE ONE OR MORE SIMULATED USER BOTS USING THE ACTION FILES CORRESPONDING TO USERS

INJECT THE ONE OR MORE SIMULATED USER BOTS INTO THE COMPUTING ENVIRONMENT (E.G., A SIMULATED USER BOT THAT EMULATES A MALICIOUS USER)

DETERMINE WHETHER A DETECTION SYSTEM DETECTS THE SIMULATED USER ACTIVITY OF THE SIMULATED USER BOT

TRANSMIT AN INDICATION INDICATING THE EFFICACY OF THE DETECTION SYSTEM (E.G., WHETHER DETECTION OCCURRED, COMPUTATIONAL COST, ETC.)

End

FIG. 6
<table>
<thead>
<tr>
<th>Threat Type</th>
<th>Behavior</th>
<th>Indicator</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals with abnormal work habits</td>
<td>Uses a work-owned machine outside of normal work hours (i.e., 12AM-7AM EST) at work, at home, or at remote sites</td>
<td>Has anti-virus updates between 12AM and 7AM EST&lt;br&gt;In the top 5% of the frequency distribution of VPN activity between 12AM and 7AM EST&lt;br&gt;In the top 5% of the frequency distribution of workstation logins between 12AM and 7AM EST&lt;br&gt;In the top 5% of the frequency distribution of email activity between 12AM and 7AM EST&lt;br&gt;In the top 5% of the frequency distribution of website activity between 12AM and 7AM EST</td>
<td>At least 1 anti-virus log entry for definition updates between 12AM and 7AM EST&lt;br&gt;At least 28 VPN log records showing a connection that started between 12AM and 7AM EST&lt;br&gt;At least 128 ADDC log entries with timestamps between 12AM and 7AM EST&lt;br&gt;At least 24 email log records for outbound emails sent between 12AM and 7AM EST&lt;br&gt;At least 24,500 proxy log entries with a timestamp between 12AM and 7AM EST</td>
</tr>
<tr>
<td>Individuals with abnormally large data transfers that are not easily explained</td>
<td>Uses a work-owned machine for abnormally large inbound or outbound data transfers</td>
<td>In the top 5% of the frequency distribution of attempted access to prohibited file sharing websites&lt;br&gt;In the top 5% of the frequency distribution of large emails sent&lt;br&gt;In the top 5% of the frequency distribution of VPN sessions that transfer large amounts of data&lt;br&gt;In the top 5% of the frequency distribution of firewall log entries that transfer large amounts of data&lt;br&gt;In the top 5% of the frequency distribution of web browsing sessions that transfer large amounts of data&lt;br&gt;In the top 5% of the frequency distribution of USB attachments</td>
<td>At least 2000 proxy log entries of attempted access to prohibited file sharing websites&lt;br&gt;At least 3 emails with attachments larger than 5 MB&lt;br&gt;At least 7 VPN sessions where at least 210 MB are transferred&lt;br&gt;At least 3 firewall log entries where at least 200 KB is transferred&lt;br&gt;At least 1 proxy log entry where at least 51 MB is transferred&lt;br&gt;At least 11 application while listing log entries of devices attached or mounted</td>
</tr>
<tr>
<td>Individuals that have unusual contact with foreign entities</td>
<td>Digitally communicates with non-US entities through a work-owned machine</td>
<td>In the top 5% of the frequency distribution of outbound emails to non-US email addresses&lt;br&gt;Have VPN sessions from non-US locations&lt;br&gt;In the top 5% of the frequency distribution of firewall log entries to non-US locations&lt;br&gt;In the top 5% of the frequency distribution of web browsing sessions to non-US websites</td>
<td>At least 1 outbound email to a non-US email address&lt;br&gt;At least 1 VPN session initiated from a non-US location&lt;br&gt;At least 37 firewall log entries to non-US locations&lt;br&gt;At least 7100 proxy log entries to non-US owned websites</td>
</tr>
</tbody>
</table>

**FIG. 8**