



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

(21) International Application Number:

PCT/US2014/063239

(22) International Filing Date:

30 October 2014 (30.10.2014)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

61/899,468 4 November 2013 (04.11.2013) US  
62/066,835 21 October 2014 (21.10.2014) US

(71) Applicant: **ILLUMIO, INC.** [US/US]; 160 San Gabriel Drive, Sunnyvale, CA 94086 (US).

(72) Inventors: **KIRNER, Paul, J.**; C/o Illumio, Inc., 160 San Gabriel Drive, Sunnyvale, CA 94086 (US). **GLENN, Matthew, K.**; C/o Illumio, Inc., 160 San Gabriel Drive, Sunnyvale, CA 94086 (US). **GUPTA, Mukesh**; C/o Illumio, Inc., 160 San Gabriel Drive, Sunnyvale, CA 94086 (US). **NAKASHIMA, Roy, N.**; C/o Illumio, Inc., 160 San

Gabriel Drive, Sunnyvale, CA 94086 (US). **VERGHESE, Thukalan, V.**; C/o Illumio, Inc., 160 San Gabriel Drive, Sunnyvale, CA 94086 (US).

(74) Agents: **HOFFMAN, Brian, M.** et al.; FENWICK & WEST LLP, Silicon Valley Center, 801 California Street, Mountain View, CA 94041 (US).

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

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,

[Continued on next page]

(54) Title: AUTOMATED GENERATION OF LABEL-BASED ACCESS CONTROL RULES.

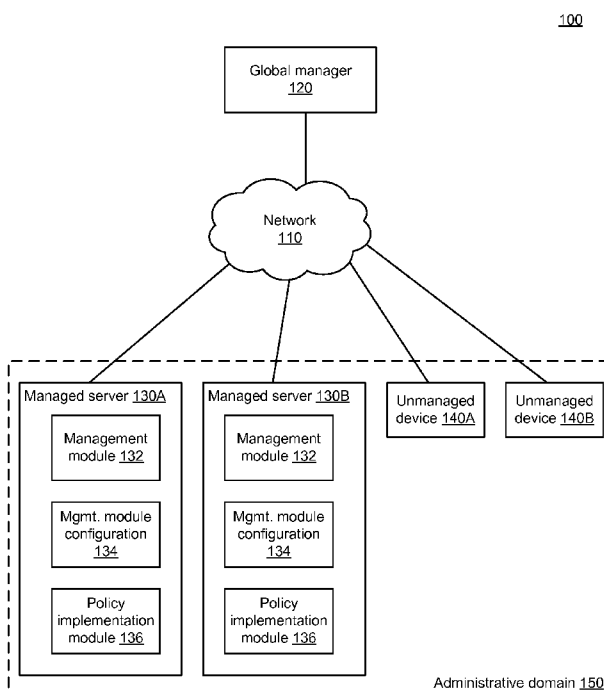


FIG. 1

(57) Abstract: An access control rule authorizing communication between a plurality of managed servers within an administrative domain is determined. Communication information describing past communication between the plurality of managed servers is obtained. A subset of managed servers from the plurality of managed servers is identified by grouping the plurality of managed servers based on the obtained communication information. A group-level label set is determined to associate with the subset of managed servers. Role labels are determined for managed servers in the subset of managed servers. A managed server is associated with one role label. Based on the group-level label set and the role labels, an access control rule is generated authorizing communication between a first managed server of the subset of managed servers and a second managed server. The access control rule is stored as part of an administrative domain-wide management policy.



TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published:**

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

## AUTOMATED GENERATION OF LABEL-BASED ACCESS CONTROL RULES.

## BACKGROUND

1. TECHNICAL FIELD

[0001] The subject matter described herein generally relates to the field of managing servers (physical or virtual) of an administrative domain and, in particular, to managing servers according to an administrative domain-wide policy that adheres to a logical multi-dimensional label-based policy model.

2. BACKGROUND INFORMATION

[0002] Servers (physical or virtual) of an administrative domain are managed according to a policy. For example, a security policy might specify access control and/or secure connectivity, while a resource-usage policy might specify usage of the administrative domain's computing resources (e.g., disks and/or peripherals). Conventional policies reference physical devices and are expressed in terms of low-level constructs such as Internet Protocol (IP) addresses, IP address ranges, subnetworks, and network interfaces. These low-level constructs make it difficult to write a fine-grained policy in an abstract and natural way.

## SUMMARY

[0003] The above and other issues are addressed by a method, non-transitory computer-readable storage medium, and system for determining an access control rule authorizing communication between a plurality of managed servers within an administrative domain. An embodiment of the method comprises obtaining communication information describing past communication between the plurality of managed servers. The method further comprises identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information. The method further comprises determining a group-level label set to associate with the subset of managed servers. The method further comprises determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label. The method further comprises generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the

subset of managed servers and a second managed server. The method further comprises storing the access control rule as part of an administrative domain-wide management policy.

**[0004]** An embodiment of the medium stores computer program modules executable by one or more processors to perform steps. The steps comprise obtaining communication information describing past communication between the plurality of managed servers. The steps further comprise identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information. The steps further comprise determining a group-level label set to associate with the subset of managed servers. The steps further comprise determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label. The steps further comprise generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the subset of managed servers and a second managed server. The steps further comprise storing the access control rule as part of an administrative domain-wide management policy.

**[0005]** An embodiment of the system comprises one or more processors and a non-transitory computer-readable storage medium storing computer program modules executable by the one or more processors to perform steps. The steps comprise obtaining communication information describing past communication between the plurality of managed servers. The steps further comprise identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information. The steps further comprise determining a group-level label set to associate with the subset of managed servers. The steps further comprise determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label. The steps further comprise generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the subset of managed servers and a second managed server. The steps further comprise storing the access control rule as part of an administrative domain-wide management policy.

**[0006]** The above and other issues are addressed by a method, non-transitory computer-readable storage medium, and system for processing alerts from managed servers implementing one or more access control rules. An embodiment of the method comprises obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past

communication between the first managed server and the second managed server. The method further comprises obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server. The method further comprises classifying the past communication as being legitimate or malicious based on the communication information. The method further comprises responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server. The method further comprises storing the access control rule as part of an administrative domain-wide management policy.

**[0007]** An embodiment of the medium stores computer program modules executable by one or more processors to perform steps. The steps comprise obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server. The steps further comprise obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server. The steps further comprise classifying the past communication as being legitimate or malicious based on the communication information. The steps further comprise responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server. The steps further comprise storing the access control rule as part of an administrative domain-wide management policy.

**[0008]** An embodiment of the system comprises one or more processors and a non-transitory computer-readable storage medium storing computer program modules executable by the one or more processors to perform steps. The steps comprise obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server. The steps further comprise obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server. The steps further comprise classifying the past communication as being legitimate or malicious based

on the communication information. The steps further comprise responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server. The steps further comprise storing the access control rule as part of an administrative domain-wide management policy.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a high-level block diagram illustrating an environment for managing servers (physical or virtual) of an administrative domain, according to one embodiment.

[0010] FIG. 2 is a high-level block diagram illustrating an example of a computer for use as one or more of the entities illustrated in FIG. 1, according to one embodiment.

[0011] FIG. 3 is a high-level block diagram illustrating a detailed view of a global manager, according to one embodiment.

[0012] FIG. 4 is a high-level block diagram illustrating a detailed view of a policy implementation module of a managed server, according to one embodiment.

[0013] FIG. 5 is a flowchart illustrating a method of generating management instructions for a particular managed server, according to one embodiment.

[0014] FIG. 6 is a flowchart illustrating a method of generating a configuration for a management module of a managed server, according to one embodiment.

[0015] FIG. 7 is a flowchart illustrating a method of monitoring local state of a managed server and sending local state information to a global manager, according to one embodiment.

[0016] FIG. 8 is a flowchart illustrating a method of processing a change to the state of an administrative domain's computer network infrastructure, according to one embodiment.

[0017] FIG. 9 is a high-level block diagram illustrating a detailed view of an access control rule creation module of a global manager, according to one embodiment.

[0018] FIG. 10 is a flowchart illustrating a method of generating an access control rule authorizing communication between a plurality of managed servers, according to one embodiment.

[0019] FIG. 11 is a flowchart illustrating a method of processing alerts from managed servers implementing one or more access control rules, according to one embodiment.

#### DETAILED DESCRIPTION

[0020] The Figures (FIGS.) and the following description describe certain embodiments

by way of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles described herein. Reference will now be made to several embodiments, examples of which are illustrated in the accompanying figures. It is noted that wherever practicable similar or like reference numbers may be used in the figures and may indicate similar or like functionality.

**[0021]** FIG. 1 is a high-level block diagram illustrating an environment 100 for managing servers (physical or virtual) 130 of an administrative domain 150, according to one embodiment. The administrative domain 150 can correspond to an enterprise such as, for example, a service provider, a corporation, a university, or a government agency. The environment 100 may be maintained by the enterprise itself or by a third party (e.g., a second enterprise) that helps the enterprise manage its servers 130. As shown, the environment 100 includes a network 110, a global manager 120, multiple managed servers 130, and multiple unmanaged devices 140. The multiple managed servers 130 and the multiple unmanaged devices 140 are associated with the administrative domain 150. For example, they are operated by the enterprise or by a third party (e.g., a public cloud service provider) on behalf of the enterprise. While one global manager 120, two managed servers 130, and two unmanaged devices 140 are shown in the embodiment depicted in FIG. 1 for clarity, other embodiments can have different numbers of global managers 120, managed servers 130, and/or unmanaged devices 140.

**[0022]** The network 110 represents the communication pathway between the global manager 120, the managed servers 130, and the unmanaged devices 140. In one embodiment, the network 110 uses standard communications technologies and/or protocols and can include the Internet. In another embodiment, the entities on the network 110 can use custom and/or dedicated data communications technologies.

**[0023]** A managed server 130 is a machine (physical or virtual) that implements an administrative domain-wide management policy 330 (shown in FIG. 3). In one embodiment, a server is a user-space instance of a virtual server (sometimes referred to as a container, virtualization engine, virtual private server, or jail) according to operating system-level virtualization, which is a server virtualization method where the kernel of an operating system enables multiple isolated user-space instances, instead of only one instance. If a managed server 130 is a physical machine, then the managed server 130 is a computer or set of computers. If a managed server 130 is a virtual machine, then the managed server 130

executes on a computer or set of computers. The administrative domain-wide management policy 330 specifies whether and/or how entities associated with the administrative domain 150 are allowed to access (or be accessed by) other entities or otherwise consume (or provide) services. For example, the administrative domain-wide management policy 330 specifies security or resource usage. A security policy might specify access control, secure connectivity, disk encryption, and/or control of executable processes, while a resource-usage policy might specify usage of the administrative domain's computing resources (e.g., disks, peripherals, and/or bandwidth).

**[0024]** A managed server 130 includes a management module 132, a management module configuration 134, and a policy implementation module 136. The management module 132 implements the administrative domain-wide management policy 330. For example, in the case of security, the management module 132 can be a low-level network or security engine such as an operating system-level firewall, an Internet Protocol security (IPsec) engine, or a network traffic filtering engine (e.g., based on the Windows Filtering Platform (WFP) development platform). In the case of resource usage, the management module 132 can be a disk-usage engine or a peripheral-usage engine.

**[0025]** The management module configuration 134 affects the operation of the management module 132. For example, in the case of security, the management module configuration 134 can be access control rules applied by a firewall, secure connectivity policies applied by an IPsec engine (e.g., embodied as iptables entries and ipset entries in the Linux operating system), or filtering rules applied by a filtering engine. In the case of resource usage, the management module configuration 134 can be disk-usage policies applied by a disk-usage engine or peripheral-usage policies applied by a peripheral-usage engine.

**[0026]** The policy implementation module 136 generates the management module configuration 134 based on a) management instructions received from the global manager 120 and b) the state of the managed server 130. The management instructions are generated based, in part, on the administrative domain-wide management policy 330. The management module configuration 134 generated by the policy implementation module 136 implements that administrative domain-wide management policy 330 (to the extent that the policy concerns the managed server 130). This two-step process (generating management instructions and generating the management module configuration 134) is referred to as "instantiating" a management policy. The policy implementation module 136 also monitors the local state of the managed server 130 and sends local state information to the global



manager 120.

**[0027]** In one embodiment, the policy implementation module 136 is part of a larger proprietary module (not shown). The proprietary module is loaded onto a device (or virtual device) that already has a management module 132 and a management module configuration 134, thereby transforming the device (or virtual device) from an unmanaged device 140 to a managed server 130. The policy implementation module 136 is further described below with reference to FIGS. 4, 6, and 7.

**[0028]** An unmanaged device 140 is a computer (or set of computers) that does not include a policy implementation module 136. An unmanaged device 140 does not implement the administrative domain-wide management policy 330. However, interaction between a managed server 130 and an unmanaged device 140 can be subject to the administrative domain-wide management policy 330 (as implemented by the managed server 130). One example of an unmanaged device 140 is a network circuit that is used by an administrative domain 150. Another example of an unmanaged device 140 is a device used by a person to authenticate himself to the administrative domain 150 (e.g., a notebook or desktop computer, a tablet computer, or a mobile phone).

**[0029]** The global manager 120 is a computer (or set of computers) that generates management instructions for managed servers 130 and sends the generated management instructions to the servers. The management instructions are generated based on a) the state of the administrative domain's computer network infrastructure 320 and b) an administrative domain-wide management policy 330. The state of the administrative domain's computer network infrastructure 320 includes descriptions of managed servers 130 and (optionally) descriptions of unmanaged devices 140. The global manager 120 also processes local state information received from managed servers 130.

**[0030]** The administrative domain-wide management policy 330 is based on a logical management model that can reference managed servers 130 based on their high-level characteristics, referred to herein as "labels." A label is a pair that includes a "dimension" (a high-level characteristic) and a "value" (the value of that high-level characteristic). A management policy constructed in this multi-dimensional space is more expressive than a management policy constructed according to a single-characteristic network/IP address-based policy model. In particular, expressing management policy using the higher-level abstractions of "labels" enables people to better understand, visualize, and modify management policy.

**[0031]** The logical management model (e.g., the number and types of dimensions

available and those dimensions' possible values) is configurable. In one embodiment, the logical management model includes the following dimensions and values, as shown in Table 1:

Dimension	Meaning (M), Values (V)
Role	M: The role of the managed server within the administrative domain. V: web, API, database
Environment	M: The lifecycle stage of the managed server. V: production, staging, development
Application	M: The logical application (higher-level grouping of managed servers) to which the managed server belongs. V: trading, human resources
Line of Business	M: The business unit to which the managed server belongs. V: marketing, engineering
Location	M: The location of the managed server. Can be physical (e.g., country or geographical region) or logical (e.g., network). Physical is particularly useful for expressing geographic compliance requirements. V: US or EU (physical), us-west-1 or us-east-2 (logical)

Table 1 – Example of logical management model

[0032] The logical management model enables multiple managed servers 130 to be grouped together by specifying one or more labels (referred to herein as a “label set”) that describe all of the managed servers 130 in the group. A label set includes either zero values or one value for a dimension in the logical management model. A label set need not include labels for all dimensions in the logical management model. In this way, the logical management model enables the segmentation and separation of an administrative domain's managed servers 130 and the creation of arbitrary groupings of managed servers 130. The logical management model also allows for a single managed server 130 to exist in multiple

overlapping sets (i.e., multiple overlapping groups of managed servers). The logical management model does not limit the single managed server 130 to existing in a hierarchy of nested sets.

[0033] For example, in the case of security, segmentation can be used with access control policies to define groups of managed servers 130 that are subject to particular policies. Similarly, segmentation can be used with secure connectivity policies to define groups of managed servers 130 and the policies that apply to intra-group communications and inter-group communications. So, communications among a first group of managed servers 130 (specified by a first label set) can be restricted to a first secure connection setting (e.g., secure connection not required), and communications between the first group of managed servers and a second group of managed servers (specified by a second label set) can be restricted to a second secure connection setting (e.g., IPsec Encapsulating Security Payload (ESP)/Authentication Header (AH) Advanced Encryption Standard (AES)/Secure Hash Algorithm-2 (SHA-2)).

[0034] Each managed server 130 in the environment 100 implements the administrative domain-wide management policy 330 (to the extent that the policy concerns the managed server 130). As a result, the administrative domain-wide management policy 330 is applied in a distributed fashion throughout the administrative domain 150, and there are no choke points. Also, the administrative domain-wide management policy 330 is applied at the logical level independent of the administrative domain's physical network topology and network addressing schemes.

[0035] The global manager 120, the state of the administrative domain's computer network infrastructure 320, and the administrative domain-wide management policy 330 are further described below with reference to FIGS. 3, 5, and 8–11.

## COMPUTER

[0036] FIG. 2 is a high-level block diagram illustrating an example of a computer 200 for use as one or more of the entities illustrated in FIG. 1, according to one embodiment. Illustrated are at least one processor 202 coupled to a chipset 204. The chipset 204 includes a memory controller hub 220 and an input/output (I/O) controller hub 222. A memory 206 and a graphics adapter 212 are coupled to the memory controller hub 220, and a display device 218 is coupled to the graphics adapter 212. A storage device 208, keyboard 210, pointing device 214, and network adapter 216 are coupled to the I/O controller hub 222. Other embodiments of the computer 200 have different architectures. For example, the memory 206

is directly coupled to the processor 202 in some embodiments.

[0037] The storage device 208 includes one or more non-transitory computer-readable storage media such as a hard drive, compact disk read-only memory (CD-ROM), DVD, or a solid-state memory device. The memory 206 holds instructions and data used by the processor 202. The pointing device 214 is used in combination with the keyboard 210 to input data into the computer system 200. The graphics adapter 212 displays images and other information on the display device 218. In some embodiments, the display device 218 includes a touch screen capability for receiving user input and selections. The network adapter 216 couples the computer system 200 to the network 110. Some embodiments of the computer 200 have different and/or other components than those shown in FIG. 2. For example, the global manager 120 and/or the managed server 130 can be formed of multiple blade servers and lack a display device, keyboard, and other components, while the unmanaged device 140 can be a notebook or desktop computer, a tablet computer, or a mobile phone.

[0038] The computer 200 is adapted to execute computer program modules for providing functionality described herein. As used herein, the term “module” refers to computer program instructions and/or other logic used to provide the specified functionality. Thus, a module can be implemented in hardware, firmware, and/or software. In one embodiment, program modules formed of executable computer program instructions are stored on the storage device 208, loaded into the memory 206, and executed by the processor 202.

### GLOBAL MANAGER

[0039] FIG. 3 is a high-level block diagram illustrating a detailed view of a global manager 120, according to one embodiment. The global manager 120 includes a repository 300 and a processing server 310. The repository 300 is a computer (or set of computers) that stores the state of the administrative domain’s computer network infrastructure 320 and the administrative domain-wide management policy 330. In one embodiment, the repository 300 includes a server that provides the processing server 310 access to the administrative domain state 320 and the management policy 330 in response to requests.

### ADMINISTRATIVE DOMAIN STATE

[0040] The state of the administrative domain’s computer network infrastructure 320 includes descriptions of managed servers 130 and (optionally) descriptions of unmanaged devices 140. A description of a managed server 130 includes, for example, a unique identifier

(UID), an online/offline indicator, one or more configured characteristics (optional), network exposure information, service information, and one or more labels that describe the managed server 130 (a label set).

**[0041]** The UID uniquely identifies the managed server 130. The online/offline indicator indicates whether the managed server 130 is online or offline. A “configured characteristic” stores a value associated with the managed server 130 and can be any type of information (e.g., an indication of which operating system is running on the managed server). A configured characteristic is used in conjunction with a rule’s condition portion (described below).

**[0042]** The network exposure information concerns the managed server’s network interfaces. In one embodiment, the network exposure information includes, for each of the managed server’s network interfaces, an identifier of a “bidirectionally-reachable network” (BRN) to which the network interface is attached and zero or more IP addresses (and their subnets) that are used for operating within the BRN. A BRN is a set of subnets, within an organization or across organizations, where any node within the BRN can establish communication with any other node in the BRN. For example, all of the nodes in a BRN have unique IP addresses. In other words, a BRN does not contain any NATs. Network exposure information (e.g., a network interface’s BRN identifier) can be used in conjunction with a rule’s condition portion.

**[0043]** In another embodiment, the network exposure information includes routing information and/or whether the managed server is behind a network address translator (NAT) (and, if it is behind a NAT, what type of NAT – 1:1 or 1:N). The global manager 120 can determine whether a managed server 130 is behind a network address translator (NAT) (and, if it is behind a NAT, what type of NAT – 1:1 or 1:N). For example, the global manager 120 determines whether a NAT exists between the global manager 120 and the managed server 130 by comparing (a) the server’s IP address according to the TCP connection between the global manager and the server and (b) the server’s IP address according to the local state information received from the server. If (a) and (b) differ, then a NAT exists between the global manager 120 and the managed server 130. If a NAT does exist, then the global manager 120 determines the type of NAT (1:1 or 1:N) by performing data center detection. For example, the global manager 120 identifies the server’s data center by the data center’s public IP address. (Alternatively, the managed server performs data center detection by querying information that is external to the server but inside the data center. The server then

sends that information to the global manager as part of the local status.) Configuration information indicates which types of NATs are used by which data centers. If no NAT information is associated with a particular data center, then the global manager 120 assumes that the NAT type is 1:N.

**[0044]** The service information includes, for example, process information and/or package information. Process information includes, for example, names of processes that the managed server 130 is running, which network ports and network interfaces those processes are listening on, which users initiated those processes, configurations of those processes, command-line launch arguments of those processes, and dependencies of those processes (e.g., shared objects to which those processes link). (Those processes correspond to the managed server 130 providing a service or using a service.) Package information includes, for example, which packages (executables, libraries, or other components) are installed on the managed server 130, the versions of those packages, the configurations of those packages, and the hash values of those packages.

**[0045]** A description of an unmanaged device 140 includes, for example, network exposure information (e.g., the IP address of the unmanaged device 140 and an identifier of the BRN to which the unmanaged device 140 is connected). An unmanaged device 140 is part of an “unmanaged device group” (UDG). An UDG includes one or more unmanaged devices 140. For example, the “Headquarters UDG” could include the primary circuit and the backup circuit that are used by an administrative domain’s headquarters, where each circuit is associated with an IP address. An UDG is associated with a unique identifier (UID). Information stored in the administrative domain state 320 regarding an UDG includes the UID of the UDG and information regarding the unmanaged devices 140 in the UDG (e.g., their network exposure information).

**[0046]** Descriptions of managed servers 130 and unmanaged devices 140 can be loaded into the administrative domain state 320 in various ways, such as by interacting with the global manager 120 via a graphical user interface (GUI) or an application programming interface (API). Descriptions of managed servers 130 can also be loaded into the administrative domain state 320 based on local status information received from managed servers (described below).

**[0047]** Regarding managed servers’ labels specifically (and configured characteristics, if any), the assignment (or reassignment) of a value for a dimension (or the setting of a configured characteristic’s value) can be performed in even more ways. For example, the

assignment/setting can be performed using a deployment and configuration tool as part of provisioning a managed server 130. Any such tool can be used, including off-the-shelf third-party tools (e.g., Puppet Labs' Puppet software, Opscode's Chef software, or CFEngine AS' CFEngine software) and custom tools that an administrative domain 150 might have.

**[0048]** As another example, the assignment/setting can be performed by a “label/configured characteristic engine” (not shown) that calculates labels and/or configured characteristic (“CC”) values. In one embodiment, the label/CC engine calculates labels/CC values based on label/CC assignment rules. A label/CC assignment rule is a function that accesses data from the administrative domain state 320 and assigns (or suggests assignment of) a label or a CC value. A label/CC assignment rule can be preset or user-configurable. For example, the global manager 120 includes a set of predefined rules, but the end-user can modify and/or delete those rules and add new rules based on the user's own custom requirements. Label/CC assignment rules can be evaluated for a managed server 130 during the initialization process. Label/CC value suggestions can then be made for any dimension/CC, and the end-user can accept or reject those suggestions. For example, if a managed server 130 is executing the Postgres database or the MySQL database, then the suggested label could be <Role, Database>. If a managed server is executing the Linux operating system, then the suggested value for the operating system CC could be “Linux.”

**[0049]** In another embodiment, the label/CC engine calculates labels/CC values based on cluster analysis. For example, the label/CC engine uses a combination of min-cut and K-means algorithms, with additional heuristics, of connected graphs to automatically identify a cluster of highly-connected managed servers 130. The cluster of managed servers 130 might correspond to an “application” (see Table 1) in the administrative domain 150. The end-user can choose to apply a value for the Application dimension (or any other dimension) to those managed servers 130 en masse.

#### ADMINISTRATIVE DOMAIN-WIDE MANAGEMENT POLICY

**[0050]** The administrative domain-wide management policy 330 includes one or more rules. Broadly speaking, a “rule” specifies a relationship between one or more providers of a service and one or more consumers of that service.

**[0051]** Rule Function – The relationship is subjected to a “rule function”, which is the practical effect of the rule. For example, in the case of security, the rule function could be access control, secure connectivity, disk encryption, or control of executable processes. A rule with an access control function specifies whether a consumer may use a provider's

service. In one embodiment, the access control function uses a pure “whitelist” model, which means that only the allowable relationships are expressed, and all other relationships are blocked by default. A rule with a secure connectivity function specifies over what secure channels (e.g., encrypted network sessions using point-to-point data encryption) a consumer may use a provider’s service. For example, a rule with a secure connectivity function could specify that usage of a provider’s services must be encrypted when the provider is located in the US and the consumer is located in the EU. A rule with a disk encryption function specifies whether a provider must store its data on an encrypted file system. A rule with an executable process-control function specifies whether a process is allowed to execute.

**[0052]** In the case of resource usage, the rule function could be disk-usage or peripheral-usage. A rule with a disk-usage function specifies an amount of data that a consumer can store on a provider. Note that a rule can specify other rule functions as well beyond just access control, secure connectivity, disk encryption, control of executable processes, disk usage, and peripheral usage. For example, a rule function could specify which Open Systems Interconnection (OSI) model Layer-7 services to apply to network traffic, the amount of metadata to collect for security analytics, or the triggers for capturing a complete network packet. The management policy model supports any number of rule functions that can be applied.

**[0053]** A rule function can be associated with one or more settings (referred to herein as a “function profile”) that specify details regarding the practical effect of the rule. For example, settings associated with a secure connectivity rule function can be a list of cryptographic algorithms used to encrypt network traffic. In one embodiment, a rule function is associated with multiple function profiles, and a function profile includes a priority. This priority is used by the function-level instruction generation module 360, as described below.

**[0054]** Service – In general, a “service” is an arbitrary process executing on a specific network port using a specific network protocol. A service of a rule within the management policy 330 is specified by a port/protocol pair and (optionally) additional qualifications, such as process information and/or package information (described above with respect to a description of a managed server 130 within the administrative domain state 320). If a managed server 130 has multiple network interfaces, then a service can be exposed on all networks or on only a subset of those networks. The end-user specifies on which networks the service is exposed. Note that, depending on the rule function, a service might not use any network resources. For example, a service for an executable process-control rule function



does not execute on a network port using a network protocol.

**[0055]** Providers/Consumers – The one or more providers of the service and the one or more consumers (i.e., users) of the service are managed servers 130 and/or unmanaged devices 140.

**[0056]** In one embodiment, a rule is represented within the administrative domain-wide management policy 330 using a set of information that includes a rule function portion, a service portion, a provided-by portion, a used-by portion, and an optional rule condition portion. The rule function portion describes the practical effect of the rule and can be associated with one or more settings (function profiles). The service portion describes the service to which the rule applies. If the service portion indicates “All”, then the rule applies to all services.

**[0057]** The provided-by (PB) portion describes which managed servers 130 and/or unmanaged devices 140 can provide the service (i.e., who the “providers” are). If the PB portion indicates “Anybody”, then anybody (e.g., any managed server 130 or unmanaged device 140) can provide the service. If the PB portion indicates “Any managed server”, then any managed server 130 can provide the service. (“Any managed server” is equivalent to specifying a label set that contains a wildcard, thereby matching all managed servers 130.) The used-by (UB) portion describes which managed servers 130 and/or unmanaged devices 140 can use the service (i.e., who the “consumers” are). Similar to the PB portion, the UB portion can also indicate “Anybody” or “Any managed server.”

**[0058]** Within the PB portion and the UB portion, a managed server 130 is specified by using a label set (i.e., one or more labels that describe the managed server) or a UID. The ability to specify managed servers 130 using label sets stems from the logical management model, which references managed servers based on their dimensions and values (labels). An unmanaged device 140 is specified by using a UID of an unmanaged device group (UDG). If a rule specifies an UDG, then the rule includes additional information regarding the unmanaged devices 140 in that group (e.g., the devices’ network exposure information). The PB portion of a rule and/or the UB portion of a rule can include multiple items, including label sets (to specify managed servers 130), managed server UIDs, and/or UDG UIDs.

**[0059]** The rule condition portion, which is optional, specifies whether the rule applies to a particular managed server 130 and/or a particular network interface of that managed server. The rule condition portion is a Boolean expression that includes one or more configured characteristics (“CCs”; part of a managed server’s description in the

administrative domain state 320) and/or network exposure information (e.g., a network interface's BRN identifier; also part of a managed server's description in the administrative domain state 320). A CC portion of the expression specifies whether the rule applies to the particular managed server, while a network exposure information portion of the expression specifies whether the rule applies to a particular network interface of that managed server. If the expression evaluates to "true" for a particular managed server's configured characteristics (specifically, for the values of that managed server's configured characteristics) and a particular network interface's information, then the rule applies to that managed server and that managed server's relevant network interface. If the expression evaluates to "false", then the rule does not apply to that managed server and that managed server's relevant network interface. For example, if a configured characteristic stores an indication of which operating system is running on the managed server, then a rule condition portion that includes that configured characteristic can control whether the rule applies to a particular managed server based on that server's operating system.

**[0060]** Rules within the administrative domain-wide management policy 330 are organized into rule lists. Specifically, the management policy 330 includes one or more rule lists, and a rule list includes one or more rules and (optionally) one or more scopes. A "scope" constrains where (i.e., to which managed servers 130) a rule is applied. A scope includes a provided-by (PB) portion and a used-by (UB) portion that limit the application of the rules in the rule list. The PB portion of the scope limits the PB portion of the rules, and the UB portion of the scope limits the UB portion of the rules. The PB and UB portions of a scope can specify a group of managed servers 130 by using a label set. If the label set does not contain a label for a specific dimension, then there is no scoping of that dimension for the resulting group of managed servers 130. If a rule list does not include any scopes, then its rules are applied globally.

**[0061]** Different scopes can be applied to a single rule list. For example, an end-user can build a set of rules that express how the web service tier (managed servers 130 with a <Role, Web> label) consumes services from the database tier (managed servers with a <Role, Database> label), how the load-balancing tier consumes services from the web service tier, and so on. Then, if the end-user wants to apply this rule list to his production environment (managed servers 130 with an <Environment, Production> label) and to his staging environment (managed servers with an <Environment, Staging> label), he does not need to copy or duplicate the rule list. Instead, he applies multiple scopes to a single rule list (a first

scope where the PB portion and the UB portion include the <Environment, Production> label and a second scope where the PB portion and the UB portion include the <Environment, Staging> label). The scope abstraction makes the rule list scale from both a usability perspective and a computational perspective.

**[0062]** Now that the administrative domain-wide management policy 330 has been described, it is helpful to work through some examples. Consider an administrative domain 150 with a two-tier application where a user device accesses a web server (the first tier), and the web server accesses a database server (the second tier). In the first tier, the user device is the consumer, and the web server is the provider. In the second tier, the web server is the consumer, and the database server is the provider. The administrative domain 150 includes two instances of this application: one in a production environment and one in a staging environment.

**[0063]** The web servers and the database servers are managed servers 130, and their descriptions (e.g., label sets) are present in the administrative domain state 320. For example, their label sets are:

web server in production: <Role, Web> and <Environment, Production>

database server in production: <Role, Database> and <Environment, Production>

web server in staging: <Role, Web> and <Environment, Staging>

database server in staging: <Role, Database> and <Environment, Staging>

(The Application dimension, the Line of Business dimension, and the Location dimension are not relevant to this example, so their labels are omitted.)

**[0064]** Now consider the following administrative domain-wide management policy 330, which is a security policy that specifies access control and secure connectivity:

Rule List #1

- Scopes
  - <Environment, Production>
  - <Environment, Staging>
- Rules
  - #1
    - Function: Access Control
    - Service: Apache
    - PB: <Role, Web>
    - UB: Anybody

- #2
  - Function: Access Control
  - Service: PostgreSQL
  - PB: <Role, Database>
  - UB: <Role, Web>

#### Rule List #2

- Scopes: None
- Rules
  - #1
    - Function: Secure Connectivity
    - Service: All
    - PB: <Role, Database>
    - UB: Any managed server

**[0065]** Note that the rules above refer to services simply as “Apache” and “PostgreSQL” for clarity. Remember that a service is a process and is specified by a port/protocol pair and (optionally) additional qualifications, such as process information and/or package information (described above with respect to a description of a managed server 130 within the administrative domain state 320).

**[0066]** Rule List #1/Rule #1 allows any device (e.g., a user device) to connect to a web server and use the Apache service. Specifically, the allowance of a connection is specified by “Access Control” in the Function portion. The “any device” is specified by “Anybody” in the UB portion. The “web server” is specified by “<Role, Web>” (a label set that includes only one label) in the PB portion. The Apache service is specified by “Apache” in the Service portion.

**[0067]** Rule List #1/Rule #2 allows a web server to connect to PostgreSQL on a database server. Specifically, the allowance of a connection is specified by “Access Control” in the Function portion. The “web server” is specified by “<Role, Web>” in the UB portion. The “PostgreSQL” is specified by “PostgreSQL” in the Service portion. The “database server” is specified by “<Role, Database>” (a label set that includes only one label) in the PB portion.

**[0068]** Rule List #1 also prevents inter-environment connections. For example, a web server is allowed to connect to PostgreSQL on a database server if the web server and database server are both in the same environment (e.g., both in the production environment or

both in the staging environment). Both servers in the production environment is specified by “<Environment, Production>” (a label set that includes only one label) in the Scope portion, while both servers in the staging environment is specified by “<Environment, Staging>” (a label set that includes only one label) in the Scope portion. (Since the scopes in this example do not distinguish between the PB portion and the UB portion, each scope’s label set is applied to both the PB portion and the UB portion.) As a result, a web server is not allowed to connect to PostgreSQL on a database server if the servers are in different environments (e.g., if the web server is in the staging environment and the database server is in the production environment).

**[0069]** Rule List #2 states that whenever any managed server connects to a database server, that connection must be performed through an encrypted channel. Specifically, the “database server” is specified by “<Role, Database>” in the PB portion. The “encrypted channel” is specified by “Secure Connectivity” in the Function portion. The “any managed server” is specified by “Any managed server” in the UB portion. The “whenever” is specified by “All” in the Service portion.

**[0070]** Turning aside from the above example, consider the following two managed servers 130: Server 1 is a web server that is part of production, part of app1, and owned by engineering in California. It would be labeled as:

<Role, Web>

<Environment, Production>

<Application, app1>

<LB, Engineering>

<Location, US>

Server 2 is a database server that is part of production, also part of app1, and also owned by engineering but in Germany. It would be labeled as:

<Role, Database Server>

<Environment, Production>

<Application, app1>

<LB, Engineering>

<Location, EU>

**[0071]** Assume that an access control rule allows all access to all managed servers 130 that are part of app1. This rule would allow Server 1 and Server 2 to communicate with each other and would disallow a managed server 130 in Germany that is part of app2 from

communicating with Server 1 or Server 2. Now assume that a secure connectivity rule specifies that all network traffic between EU and US must be encrypted. Rule functions are independently applied. In other words, the secure connectivity rule is a separate policy that is applied independent of the access control rule. As a result, the network traffic from Server 1 to Server 2 would be allowed (given the access control rule) and encrypted (given the secure connectivity rule).

### ACCESS CONTROL RULES

**[0072]** Returning to FIG. 3, the administrative domain-wide management policy 330 includes a set of access control rules 335, which is described below in the section entitled “Access Control Rules.”

### PROCESSING SERVER

**[0073]** The processing server 310 generates management instructions for managed servers 130 and sends the generated management instructions to the servers. The processing server 310 also processes local state information received from managed servers 130. The processing server 310 includes various modules such as a policy engine module 340, a relevant rules module 350, a function-level instruction generation module 360, an actor enumeration module 370, a relevant actors module 380, an administrative domain state update module 385, and an access control rule creation module 390. In one embodiment, the processing server 310 includes a computer (or set of computers) that communicates with the repository 300 and processes data (e.g., by executing the policy engine module 340, the relevant rules module 350, the function-level instruction generation module 360, the actor enumeration module 370, the relevant actors module 380, the administrative domain state update module 385, and the access control rule creation module 390).

**[0074]** The relevant rules module 350 takes as input the administrative domain-wide management policy 330 and an indication of a particular managed server 130 (e.g., that server's UID), generates a set of rules that are relevant to that server, and outputs the set of rules. This is a filtering process by which the relevant rules module 350 examines the management policy 330 and extracts only the relevant rules for the given managed server 130. The relevant rules module 350 performs the filtering by iterating through all of the rule lists in the management policy 330, analyzing the scopes of each rule list to determine whether the scopes apply to this managed server 130 and (if the scopes do apply to this managed server 130) analyzing the rules of each rule list to determine whether those rules

apply to this managed server 130. A rule applies to a managed server 130 if a) the PB portion of the rule and/or the UB portion of the rule specifies the managed server and b) the condition portion of the rule (if present) evaluates to “true” for that managed server (specifically, for the values of that managed server’s configured characteristics and network exposure information). The end result (referred to herein as a “management policy perspective”) is a collection of two sets of rules: rules where this managed server 130 provides a service and rules where this managed server 130 consumes a service.

**[0075]** The function-level instruction generation module 360 takes as input a set of rules (e.g., a management policy perspective generated by the relevant rules module 350), generates function-level instructions, and outputs the function-level instructions. The function-level instructions are later sent to a managed server 130 as part of the management instructions. A function-level instruction is similar to a rule in that each one includes a rule function portion, a service portion, a PB portion, and a UB portion. However, whereas a rule can include multiple items within its PB portion and/or UB portion (including label sets, managed server UUIDs, and/or UDG UUIDs), a function-level instruction includes only one item within its PB portion and only one item within its UB portion. Also, whereas a rule can specify a managed server (including its multiple network interfaces) within its PB portion and/or UB portion, a function-level instruction includes only one network interface within its PB portion and UB portion.

**[0076]** The function-level instruction generation module 360 analyzes a rule and generates one or more function-level instructions based on that rule. If the rule’s PB portion includes multiple items, the rule’s UB portion includes multiple items, or a managed server referenced by the rule (in the PB portion or UB portion) has multiple network interfaces, then the function-level instruction generation module 360 generates multiple function-level instructions (e.g., one function-level instruction for each possible combination of a PB item, a UB item, and a particular network interface).

**[0077]** Consider a rule that includes two items in its PB portion (A and B) and two items in its UB portion (C and D). The function-level instruction generation module 360 would generate four function-level instructions with the following PB and UB portions: 1) PB = A, UB = C; 2) PB = A, UB = D; 3) PB = B, UB = C; 4) PB = B, UB = D. Now consider a rule that covers a managed server in its PB portion or UB portion (e.g., by specifying a UUID or a label set), and that managed server has multiple network interfaces. The function-level instruction generation module 360 would generate multiple function-level instructions (e.g.,

one function-level instruction for each network interface of the managed server).

**[0078]** The function-level instruction generation module 360 analyzes the rules, the functions within those rules, and the function profiles referenced by those rules. If a rule list includes multiple scopes, then the function-level instruction generation module 360 applies those scopes multiple times to the rule list iteratively (thereby generating a complete set of function-level instructions for each scope). Recall that a rule function can be associated with multiple function profiles, and a function profile can include a priority. The function-level instruction generation module 360 orders the rules based on the priorities of the various function profiles such that the function profile with the highest priority is used. The function-level instruction generation module 360 translates the ordered rules into function-level instructions for the managed server 130 to execute. Function-level instructions reference the appropriate managed servers 130 and/or unmanaged devices 140 (e.g., the managed servers 130 and/or unmanaged devices 140 that were referenced in the input rules), taking into account the network exposure details of the services associated with the rules.

**[0079]** Note that the function-level instruction generation module 360 can generate a function-level instruction for a particular managed server 130 that turns out to be irrelevant for that server. For example, that managed server is covered by the provided-by (PB) portion of a rule, so the function-level instruction generation module 360 generates a corresponding function-level instruction. However, the rule also includes a portion that specifies the managed server's local state (e.g., a service portion that describes the provided service). Since the global manager 120 does not know the managed server's local state (e.g., whether the managed server is actually providing that service), the generated function-level instruction is sent to the managed server. The managed server checks its local state (e.g., whether it is providing that service) and processes the function-level instruction accordingly, as explained below with reference to the policy compilation module 410.

**[0080]** The actor enumeration module 370 takes as input a collection of descriptions of managed servers 130 and unmanaged device groups (UDGs) (e.g., the state of the administrative domain's computer network infrastructure 320), generates representations of those descriptions of servers and UDGs in an enumerated form (referred to as "actor-sets"), and outputs the actor-sets. For example, the actor enumeration module 370 enumerates the managed servers 130 and the UDGs within the administrative domain state 320 and the possible label sets and assigns each a unique identifier (UID). These actor-sets can then be used in conjunction with UB portions and PB portions of rules and scopes, which specify



actors using managed server UIDs, UDG UIDs, and/or label sets.

**[0081]** Consider a logical management model that includes a set of  $N$  dimensions  $D_i$  ( $i=1, \dots, N$ ), and each dimension  $D_i$  includes a set  $S_i$  of possible values  $V_j$  ( $j=1, \dots, M_i$ ) (where the wildcard “\*” is one of the possible values). In one embodiment, the actor enumeration module 370 enumerates all label sets that are possible based on the logical management model, which are equal to the Cartesian product given by  $S_1 \times S_2 \times \dots \times S_N$ . The size of this set is  $M_1 \times M_2 \times \dots \times M_N$ . The enumeration process collapses the multi-dimensional label space of the managed servers 130 into a simple enumerated form.

**[0082]** In another embodiment, the actor enumeration module 370 enumerates only those label sets that are possible based on the administrative domain state 320 (e.g., based on descriptions of managed servers within the administrative domain 150). For example, consider a logical management model that includes 2 dimensions (X and Y), and each dimension includes 3 possible values (A, B, and \*). A managed server with the label set “<X=A>, <Y=B>” can be a member of 4 possible label sets: 1) “<X=A>, <Y=B>”, 2) “<X=A>, <Y=\*>”, 3) “<X=\*>, <Y=B>”, and 4) “<X=\*>, <Y=\*>”. Note that the managed server’s label set exists in 2-dimensional space (X and Y), while possible label sets 2, 3, and 4 are projections of the managed server’s label set into sub-dimensional spaces (label set 2 is 1-dimensional space (X), label set 3 is 1-dimensional space (Y), and label set 4 is 0-dimensional space). So, the actor enumeration module 370 enumerates those 4 possible label sets. The managed server with the label set “<X=A>, <Y=B>” cannot be a member of the label set “<X=A>, <Y=A>”, so the actor enumeration module 370 does not enumerate that label set.

**[0083]** In yet another embodiment, the actor enumeration module 370 enumerates only those label sets that are used in the administrative domain-wide management policy 330 (e.g., in UB portions and PB portions of rules and scopes).

**[0084]** An actor-set includes a UID and zero or more actor-set records. An actor-set record includes a UID (either a managed server UID or an UDG UID), an identifier of the actor’s operating system, and the IP address of the actor (managed server 130 or unmanaged device 140) given the specific BRN. For example, an actor-set might include actor-set records whose IP addresses correspond to all of the managed servers 130 covered by the label set of <Role, Database> and <Environment, Production>. As another example, an actor-set might include actor-set records whose IP addresses correspond to all of the unmanaged devices 140 in the Headquarters UDG. A single actor (e.g., managed server 130 or

unmanaged device 140) can appear in multiple actor-sets.

**[0085]** Another factor in the actor-set calculation is actors with multiple network interfaces, plus the inclusion of network topology such as network address translation (NAT). So, there could be two actor-sets for the label set of <Role, Database> and <Environment, Production>: one actor-set with the internet-facing IP addresses of those managed servers 130 (i.e., associated with a first BRN), and a different actor-set for those same managed servers with the private network-facing IP addresses of those managed servers (i.e., associated with a second BRN).

**[0086]** In one embodiment, the actor enumeration module 370 can also update actor-sets based on changes to the administrative domain state 320. For example, the actor enumeration module 370 takes as input actor-sets (previously output by the actor enumeration module) and a change to a managed server's description (within the administrative domain state 320), generates updated actor-sets (which are consistent with the changed server description), and outputs the updated actor-sets. The actor enumeration module 370 generates the updated actor-sets in different ways depending on the type of change to the managed server's description.

**[0087]** Offline/online change – If the description change indicates that the server went from online to offline, then the actor enumeration module 370 generates the updated actor-sets by removing the server's actor-set record from all input actor-sets of which the server was a member. If the description change indicates that the server went from offline to online, then the actor enumeration module 370 generates the updated actor-sets by adding the server's actor-set record to any relevant input actor-sets. (If necessary, the actor enumeration module 370 creates a new actor-set and adds the server's actor-set record to that new actor-set.)

**[0088]** Label set change – If the description change indicates that the server's label set changed, then the actor enumeration module 370 treats this like a first server (with the old label set) going offline and a second server (with the new label set) coming online.

**[0089]** Network exposure information change – If the description change indicates that the server removed a network interface, then the actor enumeration module 370 generates the updated actor-sets by removing the server's actor-set record from all input actor-sets (associated with that network interface's BRN) of which the server was a member. If the description change indicates that the server added a network interface, then the actor enumeration module 370 generates the updated actor-sets by adding the server's actor-set

record to any relevant input actor-sets (associated with that network interface's BRN). (If necessary, the actor enumeration module 370 creates a new actor-set (associated with that network interface's BRN) and adds the server's actor-set record to that new actor-set.) If the description change indicates that the server changed a network interface's BRN, then the actor enumeration module 370 treats this like a first network interface (with the old BRN) being removed and a second network interface (with the new BRN) being added. If the description change indicates that the server changed a network interface's IP address (but not the BRN), then the actor enumeration module 370 generates the updated actor-sets by modifying the server's actor-set record in all input actor-sets (associated with that network interface's BRN) of which the server was a member.

**[0090]** The relevant actors module 380 takes as input one or more actor-sets (e.g., the managed servers 130 and the UDGs within the administrative domain state 320 in enumerated form) and a set of rules (e.g., a management policy perspective), determines which actor-sets are relevant to those rules, and outputs only those actor-sets. This is a filtering process by which the relevant actors module 380 examines the actor-sets and extracts only the relevant actor-sets for the given set of rules. The relevant actors module 380 performs the filtering by iterating through all of the input actor-sets, analyzing the PB portions and UB portions of the input rules to determine whether a particular actor-set is referenced by any of the rules' PB portions or UB portions. The end result (referred to herein as an "actor perspective") is a collection of actor-sets. The actor perspective is later sent to a managed server 130 as part of the management instructions.

**[0091]** In one embodiment, the relevant actors module 380 uses the input set of rules to generate an "actor-set filter." The actor-set filter selects, from the input actor-sets, only the actor-sets that are relevant to the input rules. In other words, the relevant actors module 380 uses the actor-set filter to filter the input actor-sets into relevant actor-sets.

**[0092]** The policy engine module 340 generates management instructions for managed servers 130 and sends the generated management instructions to the servers. The policy engine module 340 generates the management instructions (using the relevant rules module 350, the function-level instruction generation module 360, the actor enumeration module 370, and the relevant actors module 380) based on a) the state of the administrative domain's computer network infrastructure 320 and b) the administrative domain-wide management policy 330.

**[0093]** For example, the policy engine module 340 executes the relevant rules module

350, providing as input the administrative domain-wide management policy 330 and the UID of a particular managed server 130. The relevant rules module 350 outputs a set of rules that are relevant to that server (a “management policy perspective”). The policy engine module 340 executes the actor enumeration module 370, providing as input the administrative domain state 320. The actor enumeration module 370 outputs a representation of the descriptions of the managed servers 130 and unmanaged device groups (UDGs) within the administrative domain state 320 in an enumerated form (“actor-sets”). The policy engine module 340 executes the function-level instruction generation module 360, providing as input the management policy perspective (output by the relevant rules module 350). The function-level instruction generation module 360 outputs function-level instructions. The policy engine module 340 executes the relevant actors module 380, providing as input the actor-sets (output by the enumeration module 370) and the management policy perspective (output by the relevant rules module 350). The relevant actors module 380 outputs only those actor-sets that are relevant to those rules (“relevant actor-sets”). The policy engine module 340 sends the function-level instructions (output by the function-level instruction generation module 360) and the relevant actor-sets (output by the relevant actors module 380) to the particular managed server 130.

**[0094]** In one embodiment, the policy engine module 340 caches information that was generated during the above process. For example, the policy engine module 340 caches, in association with the particular managed server 130, the management policy perspective, the function-level instructions, the actor-set filter, and/or the relevant actor-sets. As another example, the policy engine module 340 caches the administrative domain’s actor-sets (which are not specific to a particular managed server 130).

**[0095]** Since an administrative domain’s actor-sets are based on the administrative domain state 320, a change to the administrative domain state 320 can require a change to the administrative domain’s actor-sets. Similarly, since a managed server’s management instructions are based on the administrative domain state 320 and the administrative domain-wide management policy 330, a change to the administrative domain state 320 and/or a change to the administrative domain-wide management policy 330 can require a change to the managed server’s management instructions. In one embodiment, the policy engine module 340 can update an administrative domain’s actor-sets and/or update a managed server’s management instructions and then distribute these changes (if necessary) to managed servers 130. The cached information mentioned above helps the policy engine module 340

more efficiently update the administrative domain's actor-sets and/or the managed server's management instructions and distribute the changes.

**[0096]** In one embodiment, the policy engine module 340 updates an administrative domain's actor-sets (based on a change to the administrative domain state 320) and distributes the changes to managed servers 130 as follows: The policy engine module 340 executes the actor enumeration module 370, providing as input the cached actor-sets (previously output by the actor enumeration module) and the changed portion of the administrative domain state 320 (e.g., a changed server description). The actor enumeration module 370 outputs the updated actor-sets. In one embodiment, the policy engine module 340 then sends all of the updated actor-sets to all of the managed servers 130 within the administrative domain 150. However, that embodiment is inefficient, since not all managed servers are affected by changes to all actor-sets.

**[0097]** In another embodiment, only selected actor-sets are sent to selected servers. For example, a particular managed server is sent only those actor-sets that a) were previously sent to that server and b) have changed. The cached relevant actor-sets indicate which actor-sets were previously sent to that server (see (a) above). The policy engine module 340 compares the cached actor-sets to the updated actor-sets to determine which actor-sets have changed (see (b) above). The policy engine module 340 then computes the intersection of (a) and (b). Actor-sets in that intersection are sent to the particular managed server. In one embodiment, for even greater efficiency, actor-sets are sent in "diff" format, which describes differences between the cached actor-sets and the updated actor-sets. For example, the diff format specifies an actor-set identifier, an actor identifier (e.g., a managed server UID or an UDG UID), and an indication of whether that actor should be added to, removed from, or modified within the actor-set.

**[0098]** In yet another embodiment, two tables are maintained and used to improve efficiency. A first table associates a managed server 130 with actor-sets of which that managed server is a member. A second table associates a managed server 130 with actor-sets that are relevant to that managed server (e.g., as determined by the relevant actors module 380). In these tables, a managed server 130 is represented by, e.g., that managed server's UID, and an actor-set is represented by, e.g., that actor-set's UID. The policy engine module 340 uses the changed portion of the administrative domain state 320 (e.g., the changed server description) to determine which managed server's description changed. The policy engine module 340 uses the first table to determine which actor-sets that managed server was a

member of. Those actor-sets might change as a result of the changed server description. So, the policy engine module 340 uses the second table to determine which managed servers those actor-sets are relevant to. The policy engine module 340 performs the intersection computation described above for only those managed servers.

**[0100]** In one embodiment, the policy engine module 340 updates a managed server's management instructions (based on a change to the administrative domain state 320) and sends the updated management instructions to the managed server as follows: The policy engine module 340 executes the relevant rules module 350, providing as input the administrative domain-wide management policy 330 and the UID of the managed server 130. The relevant rules module 350 outputs a set of rules that are relevant to that server (a "management policy perspective"). The policy engine module 340 compares the management policy perspective that was just output to the cached management policy perspective to determine whether they differ. If the just-output management policy perspective and the cached management policy perspective are identical, then the policy engine module 340 takes no further action. In this situation, the previously-generated managed server's management instructions (specifically, the function-level instructions and relevant actor-sets) are consistent with the change to the administrative domain state 320 and do not need to be re-generated and re-sent to the managed server.

**[0101]** If the just-output management policy perspective and the cached management policy perspective differ, then the policy engine module 340 determines which rules should be added to the cached perspective and which rules should be removed from the cached perspective. The policy engine module 340 executes the function-level instruction generation module 360, providing as input the rules to add and the rules to remove. The function-level instruction generation module 360 outputs function-level instructions to add and function-level instructions to remove (relative to the cached function-level instructions, which were previously sent to the managed server). The policy engine module 340 instructs the managed server to add or remove the various function-level instructions, as appropriate. In one embodiment, for greater efficiency, function-level instructions are sent in "diff" format, which describes differences between the cached function-level instructions and the updated function-level instructions. For example, the diff format specifies a function-level instruction identifier and an indication of whether that function-level instruction should be added to or removed from the previously-sent function-level instructions.

**[0102]** The policy engine module 340 also executes the actor enumeration module 370,

providing as input the cached actor-sets and the changed portion of the administrative domain state 320 (e.g., the changed server description). The actor enumeration module 370 outputs the updated actor-sets. The policy engine module 340 executes the relevant actors module 380, providing as input the updated actor-sets and the just-output management policy perspective. The relevant actors module 380 outputs only those updated actor-sets that are relevant to those rules (“updated relevant actor-sets”).

**[0103]** The policy engine module 340 compares the updated relevant actor-sets to the cached relevant actor-sets to determine whether they differ. If the updated relevant actor-sets and the cached relevant actor-sets are identical, then the policy engine module 340 sends no actor-sets to the managed server. In this situation, the previously-generated relevant actor-sets are consistent with the change to the administrative domain state 320 and do not need to be re-sent to the managed server. If the updated relevant actor-sets and the cached relevant actor-sets differ, then the policy engine module 340 determines which actor-sets should be added, removed, or modified relative to the cached relevant actor-sets. The policy engine module 340 instructs the managed server to add, remove, or modify the various actor-sets, as appropriate. In one embodiment, for greater efficiency, actor-sets are sent in “diff” format, which describes differences between the cached relevant actor-sets and the updated relevant actor-sets. For example, the diff format specifies an actor-set identifier and an indication of whether that actor-set should be added to, removed from, or modified relative to the previously-sent actor-sets.

**[0104]** Recall that the policy engine module 340 can update a managed server’s management instructions (based on a change to the administrative domain-wide management policy 330) and send the updated management instructions to the managed server. A change to the management policy 330 is, for example, the addition, removal, or modification of a rule or a rule set. In one embodiment, a change to the management policy 330 is generated by interaction with the global manager 120 via a GUI or API. In another embodiment, a change to the management policy 330 is generated by an automated process within the global manager 120 (e.g., in response to a security threat detected by the global manager). The policy engine module 340 updates the managed server’s management instructions and sends the updated management instructions to the managed server in a similar way, regardless of whether there was a change to the management policy 330 or a change to the administrative domain state 320. However, there are a few differences.

**[0105]** In the case of a change to the management policy 330, the policy engine module

340 does not necessarily update management instructions for all managed servers 130. Instead, the policy engine module 340 compares the previous management policy 330 to the new management policy 330 to determine which rules should be added, removed, or modified relative to the previous management policy 330. The policy engine module 340 determines which managed servers 130 are affected by the changed rules (e.g., which managed servers are covered by a) the rules' and/or scopes' PB and/or UB portions and b) the rules' conditional portions (if any)). The policy engine module 340 executes the relevant rules module 350, providing as input the changed rules (instead of the entire new management policy 330) and the UID of the managed server 130 (for only those servers that are affected by the changed rules).

[0106] The administrative domain state update (ADSU) module 385 receives changes to the administrative domain state 320 and processes those changes. A change to the administrative domain state 320 is, for example, the addition, removal, or modification of a description of a managed server 130 (including the modification of a managed server's label set or configured characteristics) or a description of an unmanaged device or unmanaged device group. In one embodiment, a change to the administrative domain state 320 originates in local state information received from a particular managed server 130. In another embodiment, a change to the administrative domain state 320 is generated by interaction with the global manager 120 via a GUI or API. In yet another embodiment, a change to the administrative domain state 320 is generated by an automated process within the global manager 120 (e.g., in response to a security threat detected by the global manager).

[0107] For example, the ADSU module 385 receives a change regarding a particular unmanaged device 140. The ADSU module 385 stores the new information in the administrative domain state 320 (e.g., as part of an unmanaged device group of which that particular unmanaged device is a member). The ADSU module 385 then updates the administrative domain's actor-sets based on the unmanaged device group change. Specifically, the ADSU module 385 instructs the policy engine module 340 to update the administrative domain's actor-sets. In one embodiment, the ADSU module 385 waits for an event to occur before instructing the policy engine module 340 to update the administrative domain's actor-sets. This event can be, for example, receipt of a user command or occurrence of a specified maintenance window.

[0108] As another example, the ADSU module 385 receives a change regarding a particular managed server 130. The ADSU module 385 stores the new information in the



administrative domain state 320 as part of the description of that particular managed server 130. The ADSU module 385 then (optionally) analyzes that managed server's description to determine additional information regarding the server and stores that information in the description. The ADSU module 385 then determines whether to update the administrative domain's actor-sets and/or the managed server's management instructions based on a change to the managed server's description. If the ADSU module 385 determines to update the administrative domain's actor-sets, then the ADSU module 385 instructs the policy engine module 340 to update the administrative domain's actor-sets. In one embodiment, the ADSU module 385 waits for an event to occur before instructing the policy engine module 340 to update the administrative domain's actor-sets. If the ADSU module 385 determines to update the managed server's management instructions, then the ADSU module 385 instructs the policy engine module 340 to update the managed server's management instructions. In one embodiment, the ADSU module 385 waits for an event to occur before instructing the policy engine module 340 to update the managed server's management instructions. The aforementioned events can be, for example, receipt of a user command or occurrence of a specified maintenance window.

**[0109]** Whether or not the ADSU module 385 determines to update the administrative domain's actor-sets and/or the managed server's management instructions depends on the type of change to the managed server's description. In one embodiment, the ADSU module 385 makes this determination as shown in Table 2:

Type of Change	Whether to Update
Online to offline	Administrative domain's actor-sets: Yes Managed server's management instructions: No
Offline to online	Administrative domain's actor-sets: Yes Managed server's management instructions: Yes
Label set	Administrative domain's actor-sets: Yes Managed server's management instructions: Yes
Configured characteristic	Administrative domain's actor-sets: Yes Managed server's management instructions: Yes
Network exposure info	Administrative domain's actor-sets: Yes Managed server's management instructions: Yes (unless IP address is the only change)

Service info	Administrative domain's actor-sets: No Managed server's management instructions: Yes (only in specified situations)
--------------	---

Table 2 – Whether to update administrative domain's actor-sets and/or managed server's management instructions based on type of server description change

[0110] In one embodiment, the ADSU module 385 determines additional information regarding the server by executing the label/configured characteristic engine and providing the server's description as input. The label/CC engine calculates labels/CC values for the server based on the server's description and label/CC assignment rules. In another embodiment, the ADSU module 385 determines whether the server is behind a network address translator (NAT) (and, if it is behind a NAT, what type of NAT – 1:1 or 1:N).

[0111] The access control rule creation module 390 is described below in the section entitled "Access Control Rules."

#### POLICY IMPLEMENTATION MODULE

[0112] FIG. 4 is a high-level block diagram illustrating a detailed view of a policy implementation module 136 of a managed server 130, according to one embodiment. The policy implementation module 136 includes a local state repository 400, a policy compilation module 410, a local state update module 420, and an alert generation module 430. The local state repository 400 stores information regarding the local state of the managed server 130. In one embodiment, the local state repository 400 stores information regarding the managed server's operating system (OS), network exposure, and services. OS information includes, for example, an indication of which OS is running. Network exposure information and service information were described above with respect to a description of a managed server 130 within the administrative domain state 320.

[0113] The policy compilation module 410 takes as input management instructions and state of a managed server 130 and generates a management module configuration 134. For example, the management instructions are received from the global manager 120 and include function-level instructions (generated by the function-level instruction generation module 360) and relevant actor-sets (output by the relevant actors module 380). The state of the managed server 130 is retrieved from the local state repository 400. In one embodiment, execution of the policy compilation module 410 is triggered by a) the managed server powering up or coming online, b) the managed server receiving management instructions,

and/or c) the contents of the local state repository 400 changing.

[0114] The policy compilation module 410 maps the function-level instructions and relevant actor-sets into a management module configuration 134. For example, the policy compilation module 410 maps an access control function-level instruction (which contains a port and an actor-set reference) into an iptables entry and an ipset entry in the Linux operating system or a Windows Filtering Platform (WFP) rule in the Windows operating system.

[0115] The application of management policy at a managed server 130 can be affected by the local state of that server. In one embodiment, the policy compilation module 410 evaluates a condition associated with a received function-level instruction and generates the management module configuration 134 based on the result of that evaluation. For example, the policy compilation module 410 evaluates a condition that references the operating system of the managed server's peer (i.e., the other actor in the relationship) and selects function profile attributes based on the result of that evaluation, where the selected function profile attributes are expressed in the management module configuration 134.

[0116] As another example, recall that a managed server 130 can receive a function-level instruction that turns out to be irrelevant for that server. For example, the rule includes a portion that specifies the managed server's local state (e.g., a service portion that describes the provided service). Since the global manager 120 does not know the managed server's local state (e.g., whether the managed server is actually providing that service), the generated function-level instruction is sent to the managed server. The policy compilation module 410 checks the managed server's local state (e.g., determines whether the managed server is providing that service). This determination amounts to evaluating a condition that references the managed server's local state. The policy compilation module 410 processes the function-level instruction accordingly. If the policy compilation module 410 determines that the condition evaluates to "true" (e.g., the managed server is providing that service), then the policy compilation module 410 incorporates that function-level instruction into the management module configuration 134. Specifically, the policy compilation module 410 incorporates function-level instructions into the management module configuration 134 only after evaluating the associated condition (which concerns the local state of that server). If the evaluation of the condition is false, then the policy compilation module 410 does not express the function-level instructions in the management module configuration 134. The specific conditions (e.g., their nature and particular values) are extensible. In one embodiment, the

conditions are related to the definition of a “service” and include process information and/or package information (described above with respect to a description of a managed server 130 within the administrative domain state 320).

[0117] For example, consider a function-level instruction that allows access to only the Apache service inbound on port 80 (i.e., where the managed server 130 is the “provider” or endpoint). The managed server 130 expresses this function-level instruction in the management module configuration 134 to allow access on port 80 only after evaluating the associated condition, which concerns whether the application (executing on that server) that is listening on port 80 is actually Apache and not some other application (rogue or otherwise). The managed server 130 expresses this function-level instruction in the management module configuration 134 only after determining that the associated condition evaluates to “true.” If the associated condition evaluates to “false,” then the managed server 130 does not express this function-level instruction in the management module configuration 134. As a result, the network traffic is blocked.

[0118] In one embodiment, a managed server 130 monitors its outbound connections. The managed server 130 compares outbound network traffic to its internal process table to determine which processes in that table are establishing those outbound connections. The managed server 130 can enforce a rule that allows only certain processes (given a set of requirements, mentioned above as “process information”) to establish an outbound connection.

[0119] In one embodiment (not shown), the policy compilation module 410 is located at the global manager 120 instead of at the managed server 130. In that embodiment, the global manager 120 does not send management instructions to the managed server 130. Instead, the managed server 130 sends its local state to the global manager 120. After the policy compilation module 410 generates the management module configuration 134 (at the global manager 120), the management module configuration 134 is sent from the global manager 120 to the managed server 130.

[0120] The local state update (LSU) module 420 monitors the local state of the managed server 130 and sends local state information to the global manager 120. In one embodiment, the LSU module 420 determines an initial local state of the managed server 130, stores appropriate local state information in the local state repository 400, and sends that local state information to the global manager 120. The LSU module 420 determines the local state of the managed server 130 by inspecting various parts of the server’s operating system

(OS) and/or file system. For example, the LSU module 420 obtains service information from the OS' kernel tables (networking information), the OS' system tables (package information), and the file system (files and hash values). The LSU module 420 obtains network exposure information from the OS' kernel and and/or OS-level data structures.

[0121] After the LSU module 420 sends the initial local state information to the global manager 120, the LSU module monitors changes to the local state. The LSU module monitors changes by, for example, polling (e.g., performing inspections periodically) or listening (e.g., subscribing to an event stream). The LSU module 420 compares recently-obtained local state information to information already stored in the local state repository 400. If the information matches, then the LSU module 420 takes no further action (until local state information is obtained again). If they differ, then the LSU module 420 stores the recently-obtained information in the local state repository 400, executes the policy compilation module 410 to re-generate the management module configuration 134 (and re-configures the management module 132 accordingly), and notifies the global manager 120 of the change. In one embodiment, the LSU module 420 sends changes to local state information to the global manager 120 in "diff" format, which describes differences between the local state information that was previously stored in the local state repository 400 (and, therefore, previously sent to the global manager 120) and the recently-obtained local state information. For example, the diff format specifies a type of local state information (e.g., operating system) and a new value for that information type. In another embodiment, the LSU module 420 sends the entire contents of the local state repository 400 to the global manager 120.

[0122] The alert generation module 430 is described below in the section entitled "Access Control Rules."

#### GENERATING MANAGEMENT INSTRUCTIONS

[0123] FIG. 5 is a flowchart illustrating a method 500 of generating management instructions for a particular managed server 130, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1. In one embodiment, the method 500 is executed multiple times (e.g., once for each managed server 130 in an administrative domain 150).

[0124] When the method 500 starts, the state of the administrative domain's computer network infrastructure 320 and an administrative domain-wide management policy 330 have already been stored in the repository 300 of the global manager 120. At this point, the method

500 begins.

[0125] In step 510, the administrative domain state 320 and the administrative domain-wide management policy 330 are accessed. For example, the policy engine module 340 sends a request to the repository 300 and receives the administrative domain state 320 and the administrative domain-wide management policy 330 in response.

[0126] In step 520, one or more relevant rules are determined. For example, the policy engine module 340 executes the relevant rules module 350, providing as input the administrative domain-wide management policy 330 and the UID of the particular managed server 130. The relevant rules module 350 outputs a set of rules that are relevant to that server (management policy perspective).

[0127] In step 530, actors are enumerated. For example, the policy engine module 340 executes the actor enumeration module 370, providing as input the administrative domain state 320. The actor enumeration module 370 generates a representation of the managed servers 130 and unmanaged device groups (UDGs) within the administrative domain state 320 in an enumerated form (actor-sets).

[0128] In step 540, one or more function-level instructions are generated. For example, the policy engine module 340 executes the function-level instruction generation module 360, providing as input the management policy perspective (generated in step 520). The function-level instruction generation module 360 generates function-level instructions.

[0129] In step 550, one or more relevant actors is determined. For example, the policy engine module 340 executes the relevant actors module 380, providing as input the actor-sets (generated in step 530) and the management policy perspective (generated in step 520). The relevant actors module 380 outputs only those actor-sets that are relevant to those rules (relevant actor-sets).

[0130] In step 560, management instructions are sent to the particular managed server 130. For example, the policy engine module 340 sends the function-level instructions (generated in step 540) and the relevant actor-sets (generated in step 550) to the particular managed server 130.

[0131] Note that steps 520 and 540 concern generating the management policy perspective (and resulting function-level instructions) for a particular managed server 130, while steps 530 and 550 concern generating the actor perspective for that managed server. The generation of the management policy perspective and the generation of the actor perspective are minimally dependent on each other, since step 520 generates a set of rules

that is used by step 550. Even so, keeping the management policy calculations (i.e., steps 520 and 540) and the actor-set calculations (i.e., steps 530 and 550) separate enhances the scalability of the policy engine module 340. Since the management policy calculations and the actor-set calculations are kept mostly separate, they can be performed in parallel (e.g., even for the same managed server 130). In addition, perspective calculations for different managed servers 130 can also be performed in parallel. Also, if an actor changes, then only the actor-sets need to be recalculated. (The function-level instructions do not need to be recalculated.) If a rule changes, then only the function-level instructions and the relevant actor-sets need to be recalculated. (The actors do not need to be re-enumerated.)

### CONFIGURING THE MANAGEMENT MODULE

[0132] FIG. 6 is a flowchart illustrating a method 600 of generating a configuration 134 for a management module 132 of a managed server 130, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1.

[0133] When the method 600 starts, information regarding the local state of the managed server 130 has already been stored in the local state repository 400 of the policy implementation module 136 in the managed server 130. At this point, the method 600 begins.

[0134] In step 610, management instructions are received from the global manager 120. For example, the policy compilation module 410 receives function-level instructions and relevant actor-sets from the global manager 120.

[0135] In step 620, the local state is accessed. For example, the policy compilation module 410 accesses information regarding the local state of the managed server 130 that is stored in the local state repository 400.

[0136] In step 630, a management module configuration 134 is generated. For example, the policy compilation module 410 takes as input the management instructions (received in step 610) and the local state (accessed in step 620) and generates a management module configuration 134.

[0137] In step 640, a management module 132 is configured. For example, the policy compilation module 410 configures the management module 132 to operate in accordance with the management module configuration 134 (generated in step 630).

## MONITORING A MANAGED SERVER

[0138] FIG. 7 is a flowchart illustrating a method 700 of monitoring local state of a managed server 130 and sending local state information to a global manager 120, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1.

[0139] When the method 700 starts, information regarding local state of the managed server 130 has already been stored in the local state repository 400 of the managed server 130. At this point, the method 700 begins.

[0140] In step 710, information regarding the current local state of the managed server 130 is determined. For example, the LSU module 420 determines the local state of the managed server 130 by inspecting various parts of the server's operating system (OS) and/or file system.

[0141] In step 720, a determination is performed regarding whether information regarding the current local state differs from information stored in the local state repository 400. For example, the LSU module 420 performs this determination. If the information does not differ, then the method proceeds to step 730 and ends. If the information does differ, then the method proceeds to step 740.

[0142] In step 740, the differing information is stored in the local state repository 400. For example, the LSU module 420 performs this step.

[0143] In step 750, the management module configuration 134 is re-generated (because the contents of the local state repository 400 have changed), and the management module 132 is re-configured accordingly. For example, the LSU module 420 executes the policy compilation module 410, which re-generates the management module configuration 134.

[0144] In step 760, the differing information is sent to the global manager 120. For example, the LSU module 420 performs this step.

## UPDATING THE ADMINISTRATIVE DOMAIN STATE

[0145] FIG. 8 is a flowchart illustrating a method 800 of processing a change to the state of an administrative domain's computer network infrastructure 320, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1.



**[0146]** In step 810, a change regarding a particular managed server 130 is received. For example, the administrative domain state update (ADSU) module 385 receives an online/offline indicator, an operating system indicator, network exposure information, and/or service information from the managed server 130 as part of local state information.

**[0147]** In step 820, the received information is stored. For example, the ADSU module 385 stores the received online/offline indicator, network exposure information, and/or service information in the administrative domain state 320 (specifically, in the description of the managed server 130 to which the information pertains).

**[0148]** In step 830, the server description is analyzed to determine additional information regarding the server. For example, the ADSU module 385 uses a label/configured characteristic engine to calculate labels/CC values for the server and/or determines whether the server is behind a network address translator (NAT) (and, if it is behind a NAT, what type of NAT – 1:1 or 1:N) and stores that information in the server description. Step 830 is optional.

**[0149]** In step 840, a determination is made regarding whether to update the administrative domain's actor-sets. For example, the ADSU module 385 determines whether to update the administrative domain's actor-sets based on a change to the managed server's description. If a determination is made to update the administrative domain's actor-sets, then the method proceeds to step 850. If a determination is made not to update the administrative domain's actor-sets, then the method proceeds to step 860.

**[0150]** In step 850, the administrative domain's actor-sets are updated. For example, the ADSU module 385 instructs the policy engine module 340 to update the administrative domain's actor-sets and notify affected managed servers 130 accordingly. In one embodiment (not shown), the ADSU module 385 waits for an event to occur before instructing the policy engine module 340 to update the administrative domain's actor-sets.

**[0151]** In step 860, a determination is made regarding whether to update the managed server's management instructions. For example, the ADSU module 385 determines whether to update the managed server's management instructions based on a change to the managed server's description. If a determination is made to update the managed server's management instructions, then the method proceeds to step 870. If a determination is made not to update the managed server's management instructions, then the method proceeds to step 880.

**[0152]** In step 870, the managed server's management instructions are updated. For example, the ADSU module 385 instructs the policy engine module 340 to update the

managed server's management instructions. In one embodiment (not shown), the ADSU module 385 waits for an event to occur before instructing the policy engine module 340 to update the managed server's management instructions.

[0153] In step 880, the method ends.

#### ACCESS CONTROL RULES

[0154] Recall that the administrative domain-wide management policy 330 of the global manager 120 includes a set of access control rules 335. The set of access control rules 335 contains one or more access control rules, which are rules with an access control rule function. Broadly, an access control rule authorizes communication between a first managed server 130 and either a second managed server 130 or an unmanaged device 140 or a device external to the administrative domain 150. In one embodiment, an access control rule specifies whether a consumer may use a provider's service. Such an access control rule specifies a provided-by (PB) portion, a used-by (UB) portion, and a service. In one embodiment, the access control rules are used in a pure "whitelist" model in which a consumer may access a service on a provider only if the set of access control rules 335 includes an access control rule with matching PB, UB, and service portions.

[0155] An access control rule may only partially specify the PB, UB, and service portions by using a wildcard in place of one or more portions. For example, if an access control rule has a UB portion that specifies a wildcard, then any managed server 130, unmanaged device 140, or other device external to the administrative domain 150 may access the service. The PB and UB portions may specify one or more particular actors (e.g., using managed server UUIDs or UDG UUIDs), one or more label sets, or a combination thereof. An example access control rule has a PB portion indicating a particular managed server 130 and a UB portion indicating the label set <Role, Database Server> and <Environment, Production>. The example access control rule allows managed servers 130 having a "Database Server" role and belonging to the "Production" environment to access the service at the particular managed server 130.

[0156] Recall that the policy implementation module 136 of a managed server 130 includes an alert generation module 430. The alert generation module 430 monitors communication (also referred to as "network traffic") between the managed server 130 and other actors (managed servers 130, unmanaged devices 140, or devices external to the administrative domain 150) for compliance with access control rules contained in the management module configuration 134. The alert generation module 430 generates an alert in

response to detecting a communication that does not comply with the access control rules (referred to as an “unauthorized communication”) and sends the alert to the global manager 120, where the alert is processed by the access control rule creation module 390 (specifically, by the alert processing module 950). An unauthorized communication includes an attempt by a consumer to use a service provided by the managed server 130 as well as an attempt by the managed server 130 to use a service provided by another actor. For example, an attempt to send network traffic to or receive network traffic from a port associated with a service can be an unauthorized communication. In an embodiment where the access control rules serve as a whitelist of permissible activities, the management module 132 allows attempted communication that matches an access control rule and denies attempted communication that does not match an access control rule.

**[0157]** When the management module 132 denies or blocks communication to or from the managed server 130, the alert generation module 430 generates an alert. The alert describes the service, the provider of the service, and the consumer of the service corresponding to the communication. The alert may contain relevant service information about the service as well as network exposure information about the provider and consumer. The alert may contain communication information that describes characteristics of the communication. Communication information may include timing, duration, frequency, protocol type, data size (e.g., total size, packet size), or data rate of the attempted communication. For example, the communication information differentiates between a single attempt to access a service and repeated attempts to access the service. Communication information may also describe routing information of communication such as source address, destination address, and path information (e.g., load balancers and NAT devices routing the unauthorized communication).

#### ACCESS CONTROL RULE CREATION MODULE

**[0158]** Recall that the processing server 310 of the global manager 120 includes an access control rule creation module 390. FIG. 9 is a high-level block diagram illustrating a detailed view of the access control rule (ACR) creation module 390 of the global manager 120, according to one embodiment. The ACR creation module 390 includes a contextual information collection module 910, a managed server grouping module 920, a labeling engine 930, a flow processing module 940, an alert processing module 950, and an access control rule (ACR) creation interface 960.

**[0159]** The contextual information collection module 910 obtains context information

describing actors in the administrative domain 150 (managed servers 130 or unmanaged devices 140) and describing communication sent or received by actors in the administrative domain 150. Context information includes managed server information, unmanaged device information, external device information, communication information, and administrative domain information.

**[0160]** Managed server information describes characteristics of a managed server 130. Managed server information includes service information such as process information and package information, as described above with respect to the administrative domain state 320. Managed server information may describe identifiers (e.g., UID, internet protocol (IP) address, media access control (MAC) address, host name), hardware resources (e.g., processor type, processor throughput, processor load, total memory, available memory, network interface devices, storage device type), or managed server type (e.g., physical device, cloud-provided virtual device, virtual machine, Linux container). Managed server information may describe software resources, such as the operating system and other software described by process information and package information.

**[0161]** A virtualized or cloud-based managed server 130 is also associated with environment information, which describes the provider of the managed server 130 (e.g., a proprietary data center, a third-party private data center, a cloud provider) as well as the communication protocol (e.g., encapsulation information, network address, network address translation) to communicate with the provider. Managed server information about a managed server 130 is stored in the managed server's local state repository 400 and sent to the global manager 120 for processing by the contextual information collection module 910. To retrieve managed server information from a virtualized or cloud-based managed server 130, the contextual information collection module 910 may query the cloud service provider or the software providing the virtual server to send managed server information or other contextual information.

**[0162]** Unmanaged device information describes characteristics of unmanaged devices 140 such as network exposure information, as described above with respect to the administrative domain state 320. Unmanaged device information may include identifiers (e.g., UDG UID, IP address, MAC address, device name), hardware resources, software resources, or network connectivity (e.g., available ports, mapping between ports and services) of an unmanaged device 140. A managed server 130 may collect unmanaged device information about unmanaged devices 140 that communicate with the managed server 130

and send the unmanaged device information to the global manager 120 for processing by the contextual information collection module 910. Alternatively or additionally, the global manager 120 queries unmanaged devices 140 in the administrative domain 150 to collect unmanaged device information. Since unmanaged devices 140 do not include a policy implementation module 136 that reports the unmanaged device's local state, unmanaged device information may be incomplete or less detailed than managed server information.

**[0163]** External device information describes characteristics of devices external to the administrative domain 150 communicating with managed servers 130. External device information may include identifiers (e.g., IP address, uniform resource locator (URL), other web address), hardware resources, software resources, or network connectivity of an external device. Managed servers 130 may collect external device information and send the information to the global manager 120 for processing by the contextual information collection module 910, but much external device information may not be visible to managed servers 130. In addition, external device information describes reputation information of the external device, which indicates trustworthiness of the external device. In one embodiment, the contextual information collection module 910 obtains reputation information matching the external device's identifier. Using the reputation information, the contextual information collection module 910 classifies the external device as safe, malicious, or neutral. Reputation information may be a binary indicator (e.g., whether the external device's identifier is on a blacklist) or a score (e.g., a relative assessment of danger associated with an identifier).

**[0164]** Communication information is described above with respect to the alert generation module 430. A managed server 130 sends communication information to the global manager 120 that describes communication sent or received by the managed server 130. In one embodiment, a managed server 130 sends communication information about communication independently of evaluating whether the communication is authorized or unauthorized. When the contextual information collection module 910 receives duplicate communication information describing the same communication, the contextual information collection module 910 may merge or de-duplicate the duplicate communication information. For example, the contextual information collection module 910 de-duplicates communication information received from two managed servers 130, one providing a service and one consuming the service.

**[0165]** The contextual information collection module 910 generates administrative domain information based on contextual information received from managed servers 130.

Administrative domain information aggregates contextual information over the administrative domain 150 or over a subset of actors in the administrative domain 150. The subset of actors in the administrative domain may be managed servers 130 described by a label set. In one embodiment, administrative domain information describes communications having at least one common characteristic. The common characteristic may be a particular port, process, protocol, or actor (e.g., a managed server 130, an unmanaged device 140, an external device). For example, the contextual information collection module 910 generates administrative domain information indicating the number of managed servers 130 having corrupted binaries associated with a particular service. As another example, the contextual information collection module 910 generates administrative domain information indicating a number of managed servers 130 scanned by a particular actor. "Scanning" refers to sending a request (e.g., probe) to a managed server 130 and using the managed server's response (or lack thereof) to obtain or automatically determine the configuration of the managed server 130 and processes executing on the managed server 130.

**[0166]** In one embodiment, the contextual information collection module 910 generates administrative domain information indicating unusual activity within the administrative domain 150. The contextual information collection module 910 identifies contextual information associated with a particular actor, a group of managed servers 130 (e.g., characterized by a common label set), a common service, or some other characteristic. The contextual information collection module 910 summarizes the contextual information using a quantity (e.g., amount of communication, number of corrupted files) and compares the quantity to a threshold quantity. The threshold quantity may be based on a preconfigured setting or may be determined dynamically based on previous historical norms for the quantity. For example, the threshold quantity is two standard deviations above the weekly moving average for the quantity. In response to the comparison to the threshold quantity, the contextual information collection module 910 determines whether the summarized contextual information is unusual. For example, the contextual information collection module 910 determines that a managed server 130 is attempting to access an unusual number of ports unassociated with any services if the number of such ports that the managed server 130 has accessed exceeds a threshold number.

**[0167]** The managed server grouping module 920 obtains communication information describing communication between actors in the administrative domain 150. Based on the communication information, the managed server grouping module 920 groups the managed

servers 130 into application groups. An application group is a set of managed servers 130 having significant volume of communication within the group compared to volume of communication with actors external to the group. In one embodiment, the managed server grouping module 920 constructs a graph where the nodes represent managed servers 130 of the administrative domain 150 and where the edges represent communication between the managed servers 130. The edges have binary values indicating presence/absence of communication between the nodes or have non-binary values quantifying the volume of communication (e.g., frequency, data size, duration). For example, the value of an edge connecting two nodes is the daily quantity of data exchanged between managed servers 130 corresponding to the two nodes. The graph may be undirected with edges that disregard direction of communication, or the graph may be directed with directed edges according to direction of communication. For example, a directional edge pointing away from a node indicates that the corresponding managed server 130 is a consumer of a service, and a directional edge pointing towards a node indicates that a managed server 130 is the provider of a service. The managed server grouping module 920 partitions the graph into sub-graphs each corresponding to an application group. For example, the managed server grouping module 920 applies a depth-first search, a k-means cluster, or a minimum cut algorithm to partition the graph. In other words, the managed server grouping module 920 groups the managed servers 130 into application groups based on communication information gathered by the contextual information collection module 910.

**[0168]** The labeling engine 930 obtains managed server information and determines labels for managed servers 130 based at least in part on the managed server information. The labeling engine 930 is similar to the labeling/CC engine but does not determine configured characteristics. In one embodiment, the labeling engine 930 determines a group-level label set (i.e., one or more group-level labels) to associate with the managed servers 130 in an application group. In one embodiment, the group-level label set includes labels with dimensions corresponding to the environment, application, and location of the managed servers 130. Labels are described further with respect to Table 1 and the administrative domain-wide management policy 330.

**[0169]** The labeling engine 930 may determine the value of a managed server's location dimension based on locations of web addresses (e.g., an IP address and/or a URL) associated with the managed server 130. The labeling engine 930 may determine the value of a managed server's label based on conditional heuristics that use contextual information (and/or

information derived from contextual information). A conditional heuristic can be created by an administrator or can be pre-configured. For example, a conditional heuristic specifies that if a managed server 130 is provided by a particular cloud service provider or located in a particular data center, then the labeling engine 930 determines a particular value for the managed server's environment dimension. As another example, a conditional heuristic specifies that if a managed server 130 contains a particular file or process (or a particular set of files or processes), then the labeling engine 930 determines a particular value for the managed server's application dimension. The labeling engine 930 may request an administrator to indicate a group-level label set or to verify an automatically generated group-level label set. The labeling engine 930 modifies the group-level label set in response to an indication or correction by the administrator.

**[0170]** Besides group-level label sets applicable to an application group, the labeling engine 930 determines role labels (i.e., labels with a role dimension) for individual managed servers 130 within an application group. In one embodiment, the labeling engine 930 determines a role label for a managed server 130 based on hardware resources, service information, or other managed server information. For example, the labeling engine 930 determines that a managed server 130 is a database if the total available memory exceeds a threshold. As another example, the labeling engine 930 determines that a managed server 130 is a load balancer based on the number of network interfaces. In one embodiment, the labeling engine 930 obtains information regarding processes executing on a managed server 130 from managed server information and determines the value of the role dimension based on the processes. Table 3 illustrates an example mapping between processes and role dimension values.

Process	Role dimension value
Postgres	Database
Oracle	Database
SQLServer	Database
Apache	HTTP server
NGINX	HTTP server
HAProxy	Load balancer

Table 3 – Mapping between processes and role dimension values



**[0171]** The flow processing module 940 obtains communication information between actors in the administrative domain 150 and generates access control rules corresponding to the communication information. In one embodiment, the flow processing module 940 identifies communication not authorized by an access control rule and generates an access control rule authorizing the communication. To generate the access control rule, the flow processing module 940 identifies the service generating the communication, the provider of the service, and the consumer of the service. The flow processing module 940 generates the access control rule with a service portion indicating the identified service, a PB portion indicating the identified provider, and a UB portion indicating the identified consumer. In one embodiment, the flow processing module 940 assumes that there are no abnormal or malicious communications in the administrative domain 150 and, accordingly, generates access control rules authorizing any communication present in the administrative domain 150.

**[0172]** In one embodiment, the flow processing module 940 generates access control rules based on group-level label sets and role labels of managed servers 130. The flow processing module 940 determines a target access control rule. For example, the target access control rule is specified by an administrator through a GUI (e.g., by indicating a particular edge of a displayed graph corresponding to the graph generated by the managed server grouping module 920). The generated access control rule specifies a service, a first managed server 130 as a provider of the service, and a second managed server 130 as a consumer of the service. The flow processing module 940 identifies role labels and group-level label sets of the first and second managed servers 130 generated by the labeling engine 930. The flow processing module 940 then generates additional access control rules applying to other consumer-provider pairs of managed servers 130 using the specified service (corresponding to particular edges of the displayed graph). The identified managed servers 130 that are providers of the service have group-level label sets and role labels matching those of the first managed server 130. The identified managed servers 130 that are consumers of the service have group-level label sets and role labels matching those of the second managed server 130. Alternatively or additionally to generating additional access control rules covering the identified consumer-provider pairs of managed servers 130, the flow processing module 940 broadens the target access control rule to include the identified consumer-provider pairs of managed servers 130. For example, the broadened access control rule's PB portion and UB portion are specified in terms of label sets including the role label and group-level label sets

rather than in terms of identifiers of particular managed servers 130.

[0173] In one embodiment, the flow processing module 940 generates an access control rule controlling communication between a first managed server 130 and another actor (e.g., an unmanaged device 140, an external device outside of the administrative domain 150). The flow processing module 940 identifies an existing access control rule specifying a service, a first managed server 130, and the other actor. The flow processing module 940 identifies a second managed server 130 having similar labels (including role label and group-level label set) as the first managed server 130. The first and second managed servers 130 are either both consumers of the specified service or both providers of the specified service. The flow processing module 940 generates another access control rule authorizing service-related communication between the second managed server 130 and the other actor. Alternatively or additionally to generating an additional access control rule, the flow processing module 940 broadens the existing access control rule by specifying the access control rule's PB portion or UB portion in terms of the first managed server's label set (including the role label and group-level label set) rather than in terms of an identifier of the first managed server 130.

[0174] In one embodiment, the flow processing module 940 generates rules to modify the server state of the managed servers 130 within the administrative domain 150. The server state determines to what extent the management modules 132 implement the access control rules. In an enforcement state, the management modules 132 block or terminate communication that is unauthorized according to the access control rules. For example, in a pure whitelist policy, the management modules 132 block or terminate communications that do not match at least one access control rule. The server states also include a build state and a test state, where the management modules 132 permit communications even if the communications are not authorized by an access control rule. To initiate a build state or test state, the flow processing module 940 generates an unrestricted access control rule with PB, UB, and service portions that specify wildcards. In other words, the unrestricted access control rule authorizes all communication because there are no restrictions on the access control rule's applicability to various services or actors. To transition to enforcement state from build state or test state, the flow processing module 940 removes the unrestricted access control rule.

[0175] The alert processing module 950 obtains alerts from managed servers 130, processes the alerts, and (if appropriate) generates access control rules based on the obtained alerts. In one embodiment, the alert processing module 950 obtains alerts from managed

servers 130 when the managed servers 130 are in an enforcement state or a test state. When a managed server 130 is in a build state, the alert processing module 950 instructs the managed server 130 not to generate alerts in response to detecting communication that is not authorized by an access control rule. When a managed server 130 is in a test state, the alert generation module 430 generates alerts indicating unauthorized traffic even though the management module 132 is not enforcing the access control rules to block the unauthorized traffic.

**[0176]** Before generating an access control rule in response to an alert, the alert processing module 950 classifies the communication that triggered the alert using obtained contextual information relevant to the alert. The contextual information includes communication information describing the communication, managed server information about any managed servers 130 sending or receiving the communication, or administrative domain information. If the alert is generated in response to communication with an external device, the contextual information includes external device information. If the alert is generated in response to communication with an unmanaged device 140, the contextual information includes unmanaged device information. The alert processing module 950 classifies the communication triggering the alert as being legitimate or malicious based on the obtained contextual information. For example, if the external device information indicates that the external device is malicious, then the communication is classified as malicious.

**[0177]** In one embodiment, the alert processing module 950 classifies communication as malicious if the administrative domain information indicates that the actor initiating the communication is associated with unusual activity. The contextual information collection module 910 may generate administrative domain information summarizing the number of alerts associated with a common characteristic such as a common actor, process, port, or protocol. If the number of alerts associated with the common characteristics exceeds a threshold number, then the contextual information collection module 910 classifies the communication as malicious. For example, if the number of alerts generated in response to traffic initiated by a managed server 130 exceeds a threshold number, then communication initiated by the managed server 130 is classified as malicious.

**[0178]** The alert processing module 950 may determine that obtained administrative domain information indicates the presence of a progressive infection. In a progressive infection, malicious software spreads across the administrative domain 150 over time. If administrative domain information indicates that the number of alerts from a first managed

server 130 exceeds a threshold, and if a second managed server 130 in communication with the first managed server 130 begins generating alerts, then the alert processing module 950 determines that the alerts are associated with a progressive infection. Accordingly, the alert processing module 950 classifies the communication triggering alerts as malicious.

**[0179]** Alternatively or additionally to classifying the alert according to contextual information, the alert processing module 950 notifies an administrator in response to receiving the alert. Notifying the administrator may include reporting contextual information related to the communication triggering the alert. The alert processing module 950 may receive a classification from the administrator indicating whether the corresponding communication is legitimate or malicious.

**[0180]** The alert processing module 950 processes an alert according to the classification of the corresponding communication. If the corresponding communication is classified as malicious, the alert processing module 950 does not generate an access control rule authorizing the corresponding communication. In some embodiments, the alert processing module 950 instructs the managed servers 130 to cease communication with the originating actor that initiated the communication triggering the alert. In other words, the originating actor is quarantined. The alert processing module 950 notifies an administrator about the alert in response to classifying the corresponding communication as malicious. Alternatively or additionally, the alert processing module 950 notifies an administrator about the alert regardless of the alert's classification. If the corresponding communication is classified as legitimate, then the alert processing module 950 may instruct the flow processing module 940 to generate an access control rule authorizing the communication. In some embodiments, the alert processing module 950 may request approval for the access control rule from an administrator before adding the access control rule to the set of access control rules 335.

**[0181]** The access control rule (ACR) creation interface 960 provides an administrator an interface for reviewing contextual information, application groups, label sets (e.g., including role labels and/or group-level label sets) assigned to managed servers 130, and access control rules. The ACR creation interface 960 may receive a corrected application group of a managed server 130 from an administrator. In response, the managed server grouping module 920 updates the managed server's application group to match the corrected application group. Additionally, the labeling engine 930 updates the group-level label set of the managed server 130 to match the group-level label set of the newly selected application

group. The ACR creation interface 960 may receive a corrected label set for a managed server 130, and the labeling engine 930 updates the managed server's label set according to the correction. In response to the administrator modifying an application's group-level label set, the labeling engine 930 modifies group-level label sets of other managed servers 130 in the application group to match the corrected group-level label set.

[0182] The ACR creation interface 960 may receive a target access control rule from an administrator (e.g., by the administrator indicating a particular edge of a displayed graph). For example, the administrator's target access control rule indicates a service, the service's provider, and the service's consumer. The flow processing module 940 generates an access control rule according to the administrator's instructions and possibly generates additional access control rules (or broadens the generated access control rule) based on the service and the label sets of the provider and consumer.

[0183] The ACR creation interface 960 may notify the administrator about alerts obtained by the alert processing module 950. The ACR creation interface 960 may receive a classification of the communication triggering the alert, and the flow processing module 940 may generate an access control rule according to the classification. In one embodiment, the ACR creation interface 960 presents an administrator with an access control rule automatically generated by the flow processing module 940. The ACR creation interface 960 may receive the administrator's approval, modification, or denial of the auto-generated access control rule. The flow processing module 940 adds the (possibly modified) auto-generated access control rule to the set of access control rules 335 in response to receiving approval or modification from an administrator.

#### GENERATING ACCESS CONTROL RULES

[0184] FIG. 10 is a flowchart illustrating a method 1000 of generating an access control rule authorizing communication between a plurality of managed servers 130, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1.

[0185] In step 1010, communication information describing past communication between the plurality of managed servers 130 is obtained. For example, the communication information describes the daily amount of data transferred between each pair of the managed servers 130. Step 1010 is performed by, for example, the contextual information collection module 910.

[0186] In step 1020, a subset of managed servers 130 is identified from the plurality of managed servers 130 by grouping the plurality of managed servers 130 based on the obtained communication information. For example, the subset is determined by applying a k-means clustering algorithm to a graph with nodes representing the managed servers 130 and edges having values reflecting daily amount of data transferred between pairs of managed servers 130. Step 1020 is performed by, for example, the managed server grouping module 920.

[0187] In step 1030, a group-level label set is determined to associate with the subset of managed servers 130. For example, the label set includes an application label (e.g., <Application, Human Resources>), a location label (e.g., <Location, North America>), and an environment label (e.g., <Environment, Production>). Step 1030 is performed by, for example, the labeling engine 930.

[0188] In step 1040, role labels are determined for managed servers 130 in the subset of managed servers. A managed server 130 is associated with one role label. For example, a first managed server 130 is associated with a role label having a “Database” value, and a second managed server 130 is associated with a role label having a “Web Server” value based on processes executing on the respective managed servers 130. Step 1040 is performed by, for example, the labeling engine 930.

[0189] In step 1050, an access control rule authorizing communication between a first managed server 130 of the subset of managed servers 130 and a second managed server 130 is generated based on the group-level label set and the role labels. The second server 130 may be part of the subset of managed servers 130 or part of another subset of managed servers 130. For example, the PB portion of the access control rule indicates that the first managed server 130 is the provider of an “sshd” (ssh daemon) service, and the UB portion of the access control rule indicates that the second managed server 130 is the consumer of the “sshd” service. Step 1050 is performed by, for example, the flow processing module 940.

[0190] In step 1060, the access control rule is stored as part of the set of access control rules 335. Step 1060 is performed by, for example, the flow processing module 940.

[0191] In step 1070, the method ends. Later, the policy engine module 340 processes the change to the administrative domain-wide management policy 330. The processing results in translating the access control rule to function-level instructions for one or more relevant managed servers 130 to implement the access control rule and sending the function-level instructions to the relevant managed servers 130.

[0192] Alternatively or additionally to generating access control rules, the methods

described herein may be used to facilitate creation of other rules with different rule functions as part of the administrative domain-wide management policy 330. Some rules specify both the provider of a service and a consumer of a service. One such example rule has a secure connectivity function specifying protocols, encryption, or channels to be used with communications for a service. For these rules, the global manager 120 obtains a target rule and identifies a label set (e.g., including a role label and/or group-level labels) describing the provider and a label set describing the consumer. The global manager 120 then generates additional rules (or broadens existing rules) that apply to provider-consumer pairs with respective label set pairs that match the pair of identified label sets. The additional (or broadened) rules apply to the same service and have the same function profile (e.g., encryption protocol, communication protocol type) as the target rule.

**[0193]** Some rules specify only the provider of the service or only the consumer of the service. Example rules that specify one of a consumer or a provider may have rule functions regulating stored-data encryption, disk usage, peripheral usage, or processor usage. For these rules, the global manager 120 obtains a target rule and identifies a label set corresponding to the provider or the consumer. For rules that specify a provider, the global manager 120 generates additional rules (or broadens existing rules) that apply to providers of the service having label sets that match the identified label set. For rules that specify a consumer, the global manager 120 generates additional rules (or broadens existing rules) that apply to consumers of the service having label sets that match the identified label set. The additional (or broadened) rules apply to the same service and have the same function profile (e.g., encryption protocol, resource usage limits) as the target rule.

**[0194]** Some rules affect a managed server 130 regardless of the services provided by or consumed by the managed server 130. Example rules regulate which processes may execute on a managed server 130, general disk-encryption settings, or when to capture a network packet for security analysis. The global manager 120 obtains a target rule, identifies a label set from the target rule, and generates (or broadens) rules applying to additional managed servers 130 with label sets matching the identified label set. The additional (or broadened) rules have the same function profile as the target rule. This process is similar to that described previously except the generated rule does not specify a service.

**[0195]** In some embodiments, the flow processing module 940 generates rules based on a different class of labels than are used for other rules (e.g., access control rules). Such rules affect a service provided by or used by a managed server 130 and may be generated based on

one or more alternative or additional labels for the managed server. The labeling engine 930 may determine multiple process-specific role labels to apply to processes of a managed server 130. In one embodiment, the flow processing module 940 generates rules based on alternative role labels for the provider or the consumer of the service. The alternative role labels are the process-specific role labels associated with the one or more processes used by the managed server 130 to provide or consume the service specified by the rule.

#### PROCESSING ALERTS FROM MANAGED SERVERS

**[0196]** FIG. 11 is a flowchart illustrating a method 1100 of processing alerts from managed servers 130 implementing one or more access control rules, according to one embodiment. Other embodiments can perform the steps in different orders and can include different and/or additional steps. In addition, some or all of the steps can be performed by entities other than those shown in FIG. 1.

**[0197]** In step 1110, an alert is obtained from a first managed server 130 configured to generate the alert in response to communication with a second managed server 130. The alert is generated in response to the first managed server 130 determining that the one or more access control rules implemented by the first managed server do not authorize the communication between the first managed server 130 and the second managed server 130.

**[0198]** In step 1120, contextual information relevant to at least one of the first managed server 130, the second managed server 130, and the alert is obtained. For example, the contextual information is administrative domain information indicating the number of ports that a first managed server 130 has requested to connect to at a second managed server 130, where the second managed server 130 does not have any processes listening to the ports. As another example, the contextual information is communication information indicating the frequency of communication between the first managed server 130 and the second managed server 130.

**[0199]** In step 1130, the communication corresponding to the alert is classified as being legitimate or malicious. For example, the communication is classified as being malicious in response to the number of ports identified in the administrative domain information exceeding a threshold number of ports. As another example, the communication is classified as being legitimate in response to the frequency of communications not exceeding a threshold difference of an expected frequency of communication associated with the service.

**[0200]** In step 1140, a determination is made whether the communication is classified as legitimate. If the communication is legitimate, the method 1100 proceeds to step 1150. If



the communication is not legitimate, the method 1100 proceeds to step 1170.

**[0201]** In step 1150, an access control rule is generated permitting the communication between the first managed server 130 and the second managed server 130.

**[0202]** In step 1160, the access control rule is stored as part of a set of access control rules 335.

**[0203]** In step 1170, an administrator is notified about the alert. Notifying an administrator about an alert may include requesting the administrator approve an access control rule generated to authorize the communication corresponding to the alert if the communication is classified as being legitimate. Notifying the administrator may also include prompting the administrator to quarantine the first or second managed server 130 if the communication is classified as malicious.

**[0204]** In step 1180, the method ends. Later, the policy engine module 340 processes the change to the administrative domain-wide management policy 330. The processing results in translating the access control rule to function-level instructions for one or more relevant managed servers 130 to implement the access control rule and sending the function-level instructions to the relevant managed servers 130.

**[0205]** The above description is included to illustrate the operation of certain embodiments and is not meant to limit the scope of the invention. The scope of the invention is to be limited only by the following claims. From the above discussion, many variations will be apparent to one skilled in the relevant art that would yet be encompassed by the spirit and scope of the invention.

## CLAIMS

1. A method for determining an access control rule authorizing communication between a plurality of managed servers within an administrative domain, the method comprising:
  - obtaining communication information describing past communication between the plurality of managed servers;
  - identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information;
  - determining a group-level label set to associate with the subset of managed servers;
  - determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label;
  - generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the subset of managed servers and a second managed server; and
  - storing the access control rule as part of an administrative domain-wide management policy.
2. The method of claim 1,
  - wherein obtaining the communication information comprises identifying a service provided by the first managed server and used by the second managed server, and
  - wherein generating the access control rule comprises generating the access control rule specifying the service, the access control rule comprising a provided-by portion specifying the first managed server and a used-by portion specifying the second managed server.
3. The method of claim 1, further comprising:
  - identifying past unauthorized communication between the first managed server and a third managed server, the administrative domain-wide management policy lacking an access control rule describing the unauthorized communication;
  - determining that the unauthorized communication should be permissible based on information describing the unauthorized communication, one or more labels associated with the first managed server, and one or more labels associated with the third managed server; and

generating a second access control rule authorizing the previously-unauthorized communication.

4. The method of claim 1, further comprising:

identifying a third managed server similar to the first managed server, the first managed server and third managed server associated with matching role labels and matching group-level label sets;

identifying a fourth managed server similar to the second managed server, the second managed server and fourth managed server associated with matching role labels and matching group-level label sets; and

generating a second access control rule authorizing communication between the third managed server and the fourth managed server.

5. The method of claim 1, further comprising:

identifying a third managed server similar to the first managed server, the first managed server and third managed server associated with matching role labels and matching group-level label sets;

identifying a fourth managed server similar to the second managed server, the second managed server and fourth managed server associated with matching role labels and matching group-level label sets; and

broadening the access control rule to authorize communication between the third managed server and the fourth managed server.

6. The method of claim 1, further comprising:

generating a second access control rule authorizing communication between the first managed server and a device external to the administrative domain;

identifying a third managed server similar to the first managed server, the first managed server and third managed server associated with matching role labels and matching group-level label sets; and

generating a third access control rule authorizing communication between the third managed server and the device external to the administrative domain.

7. The method of claim 1, further comprising:

generating a second access control rule authorizing communication between the first managed server and a device external to the administrative domain;

- identifying a third managed server similar to the first managed server, the first managed server and third managed server associated with matching role labels and matching group-level label sets; and
- broadening the second access control rule to authorize communication between the third managed server and the device external to the administrative domain.
8. The method of claim 1, wherein the obtained communication information describes characteristics of data previously transferred between the plurality of managed servers, and wherein identifying the subset of managed servers comprises:
- identifying the subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the characteristics of data previously transferred between the plurality of managed servers.
9. The method of claim 1, wherein the obtained communication information describes processes executed by the subset of managed servers, and wherein determining the role labels comprises:
- determining a role label for a managed server based on one or more processes executed by the managed server.
10. The method of claim 1, wherein the obtained communication information describes hardware resources of the subset of managed servers, and wherein determining the role labels comprises:
- determining a role label for a managed server based on hardware resources of the managed server.
11. The method of claim 1, further comprising:
- requesting that an administrator verify at least one of the determined group-level label set and the determined role labels; and
- modifying at least one of the group-level label set and the role labels responsive to a correction from the administrator.
12. The method of claim 1, further comprising:
- identifying a second subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information, the second subset of managed servers comprising the second managed server.
13. A method of processing alerts from managed servers implementing one or more access control rules, the method comprising:

obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server;

obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server;

classifying the past communication as being legitimate or malicious based on the communication information;

responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server; and

storing the access control rule as part of an administrative domain-wide management policy.

14. The method of claim 13, further comprising instructing the first managed server to terminate the past communication responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server.

15. The method of claim 13, further comprising instructing the first managed server to cease all communication with the second managed server responsive to classifying the past communication as malicious.

16. The method of claim 13, further comprising instructing the first managed server to permit the past communication.

17. The method of claim 13,

wherein obtaining the contextual information comprises determining a number of managed servers generating alerts in response to past communications initiated by the second managed server, and

wherein classifying the past communication comprises classifying the past communication as being malicious responsive to the number of managed servers exceeding a threshold number of managed servers.

18. The method of claim 13,

wherein obtaining the contextual information comprises identifying one or more ports of the first managed server that the second managed server has requested to connect to, the first managed server lacking any processes listening to the identified ports, and

wherein classifying the past communication comprises classifying the past communication as being malicious responsive to a number of the identified ports exceeding a threshold number of ports.

19. The method of claim 13,

wherein obtaining the contextual information comprises identifying one or more alerts generated by the second managed server prior to the alert obtained from the first managed server, the identified one or more alerts generated in response to past communication between the second managed server and at least one additional managed server, and

wherein classifying the past communication comprises classifying the past communication as being malicious based on the one or more alerts generated by the second managed server.

20. The method of claim 13, wherein generating the access control rule comprises:

requesting that an administrator approve the access control rule.

21. A non-transitory, computer-readable storage medium stores computer program modules executable by one or more processors to perform steps for determining an access control rule authorizing communication between a plurality of managed servers within an administrative domain, the steps comprising:

obtaining communication information describing past communication between the plurality of managed servers;

identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information;

determining a group-level label set to associate with the subset of managed servers;

determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label;

generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the subset of managed servers and a second managed server; and  
storing the access control rule as part of an administrative domain-wide management policy.

22. A system for processing alerts from managed servers implementing one or more access control rules, the system comprising:

one or more processors; and

a non-transitory, computer-readable storage medium stores computer program modules executable by one or more processors to perform steps comprising:

obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server;

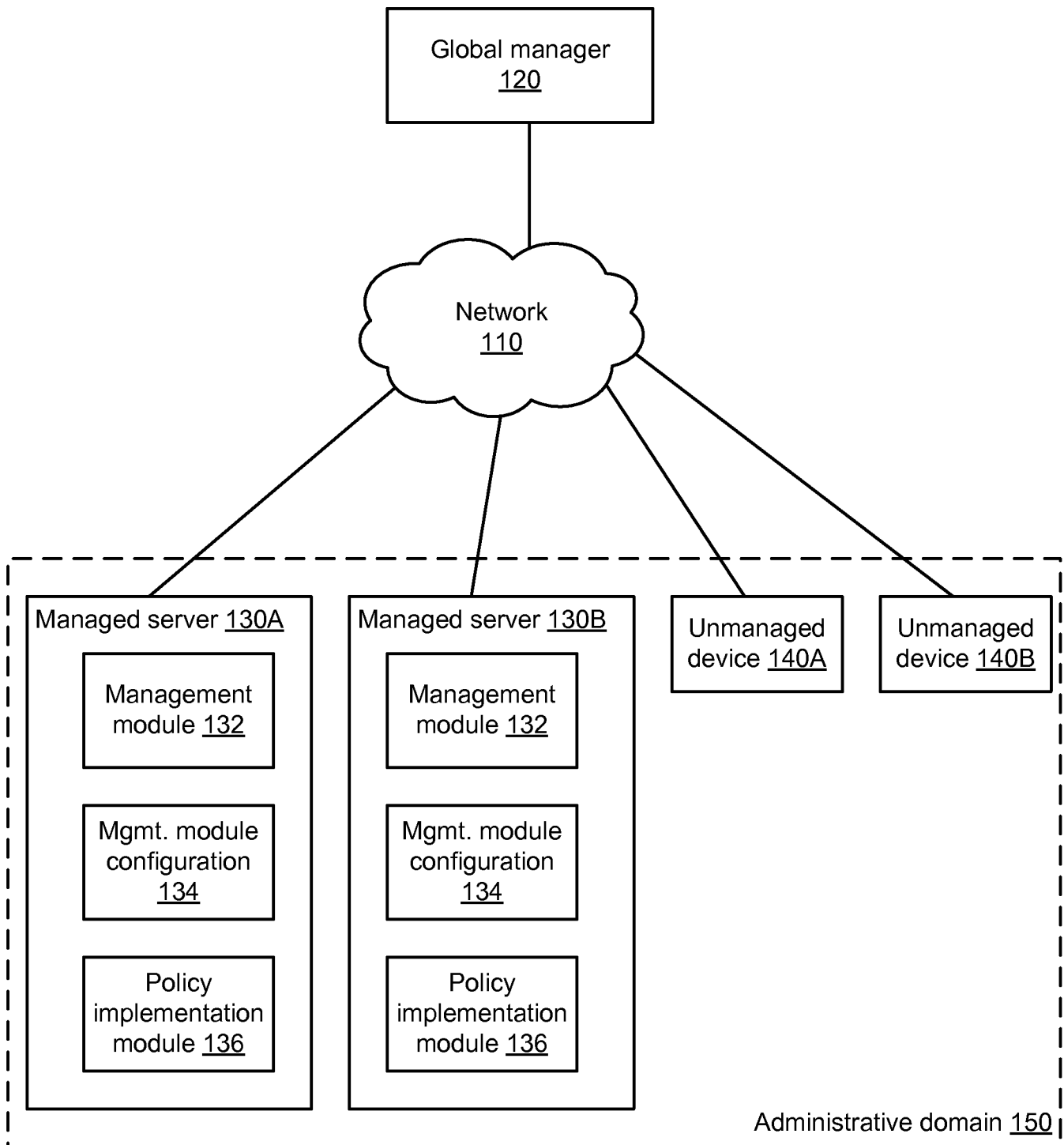
obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server;

classifying the past communication as being legitimate or malicious based on the communication information;

responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server; and

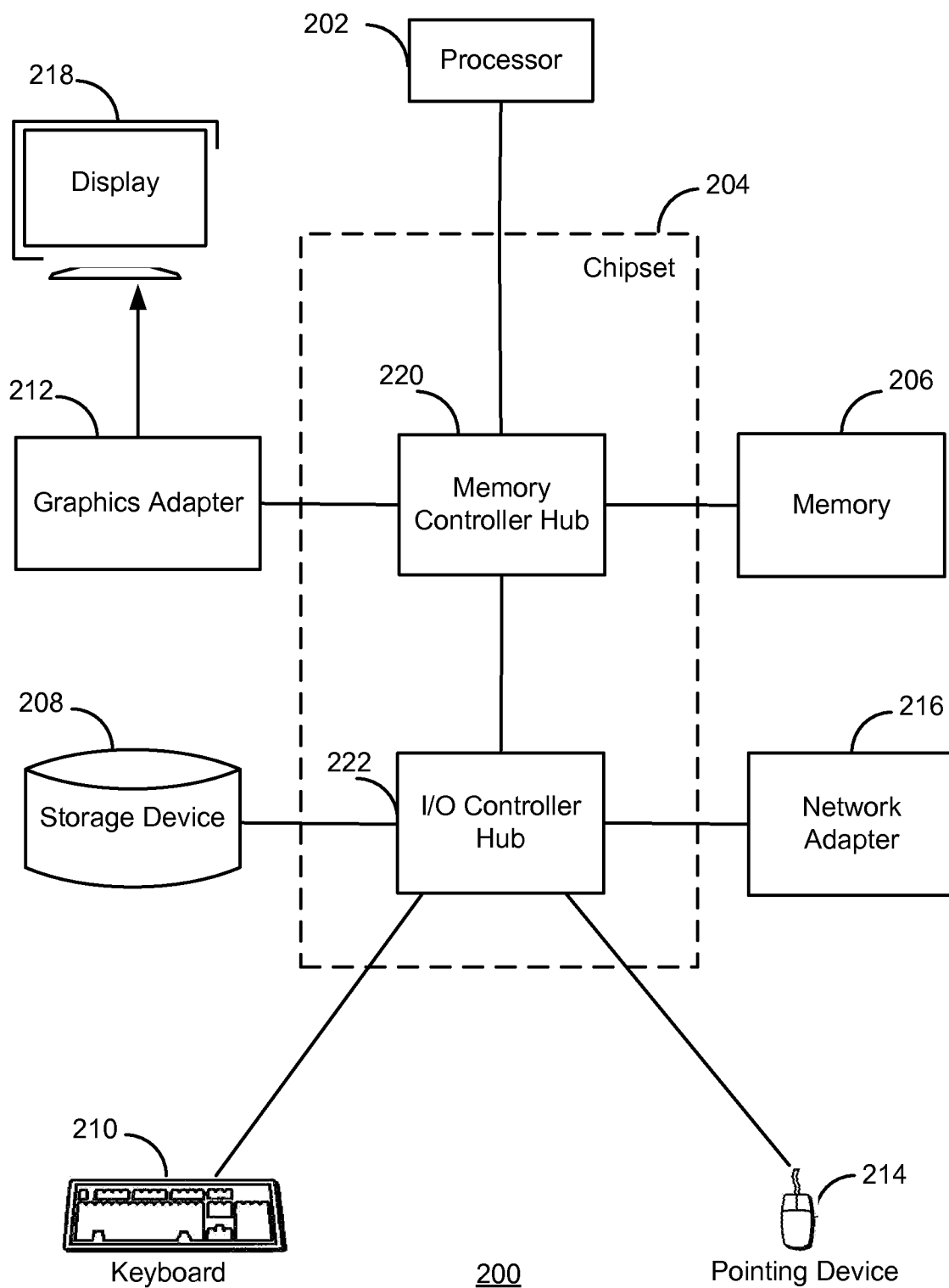
storing the access control rule as part of an administrative domain-wide management policy.

1/11

100**FIG. 1**



2/11

**FIG. 2**

3/11

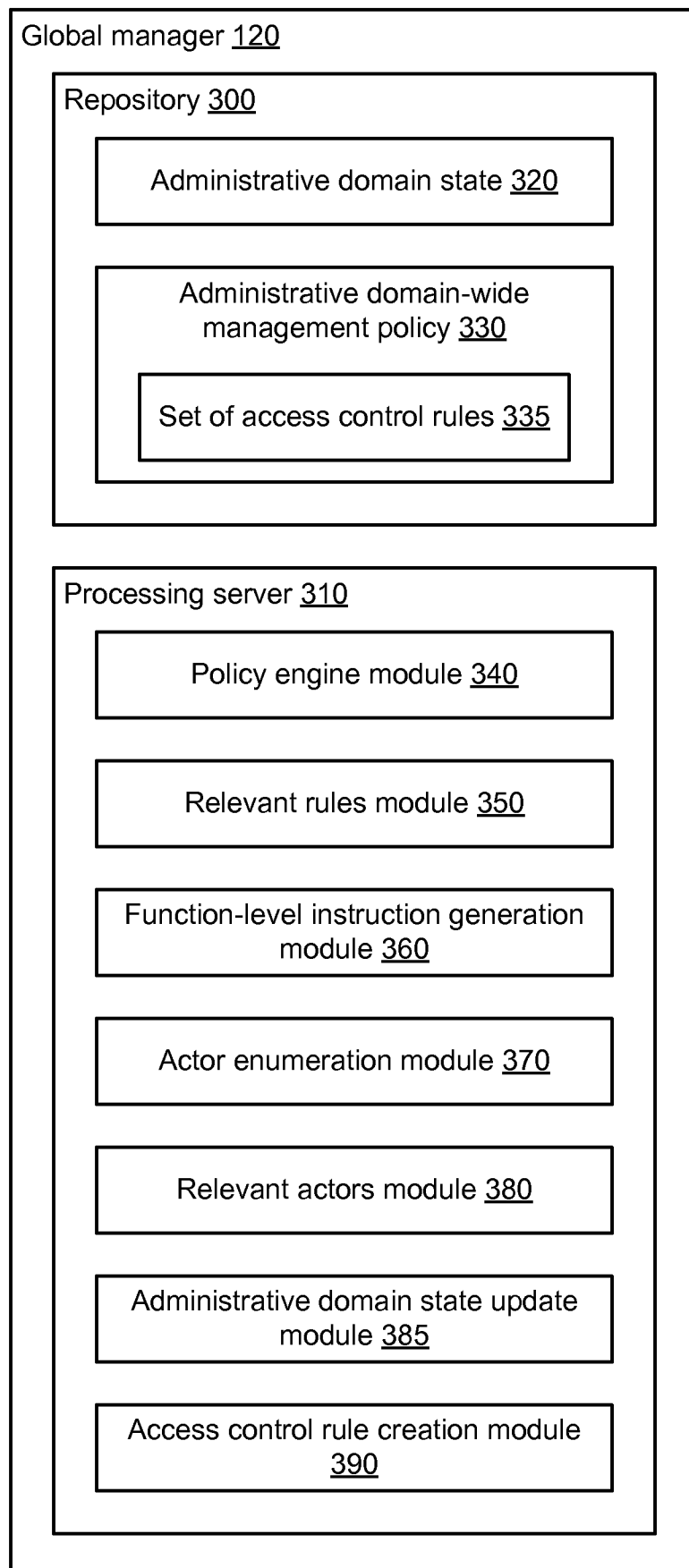
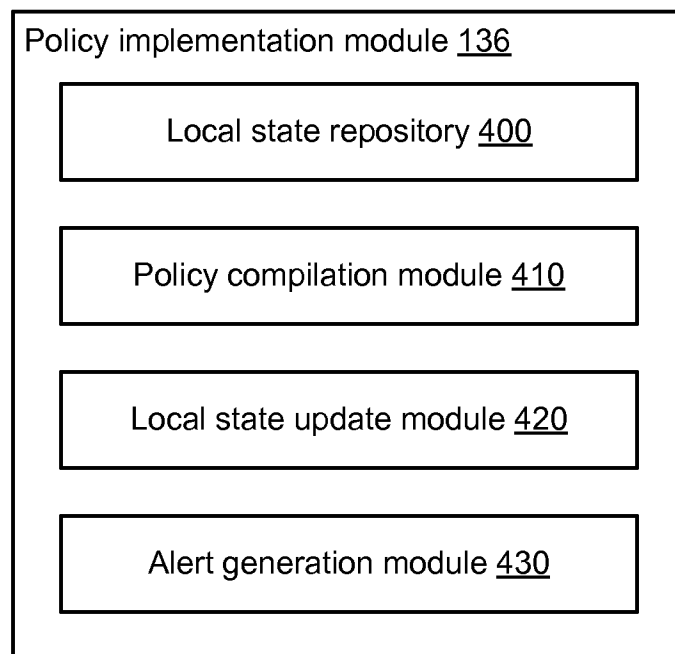
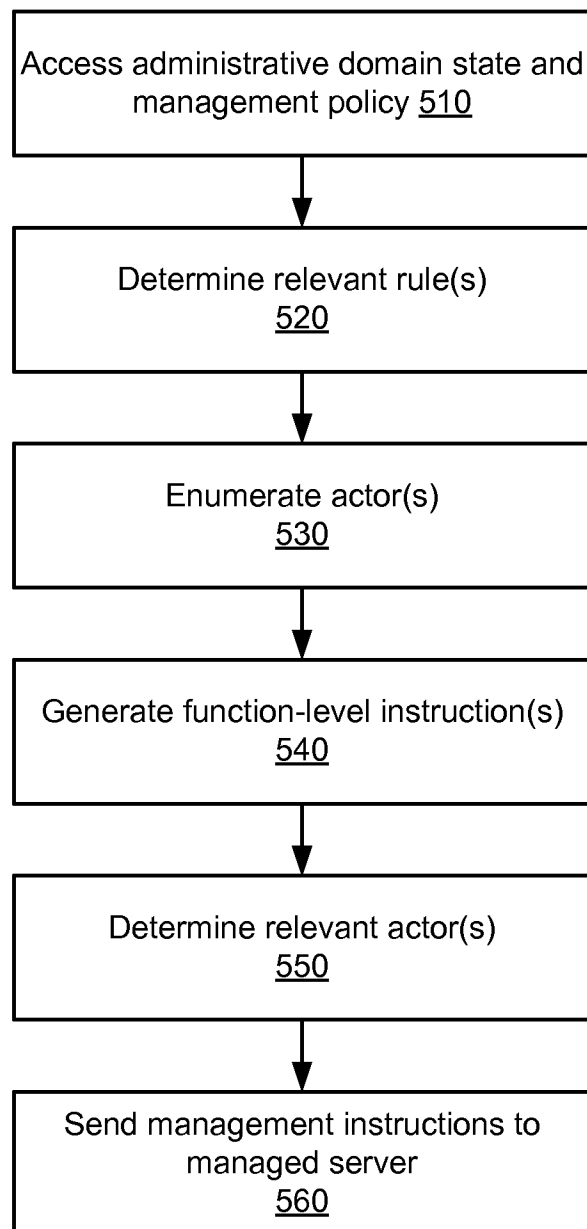


FIG. 3

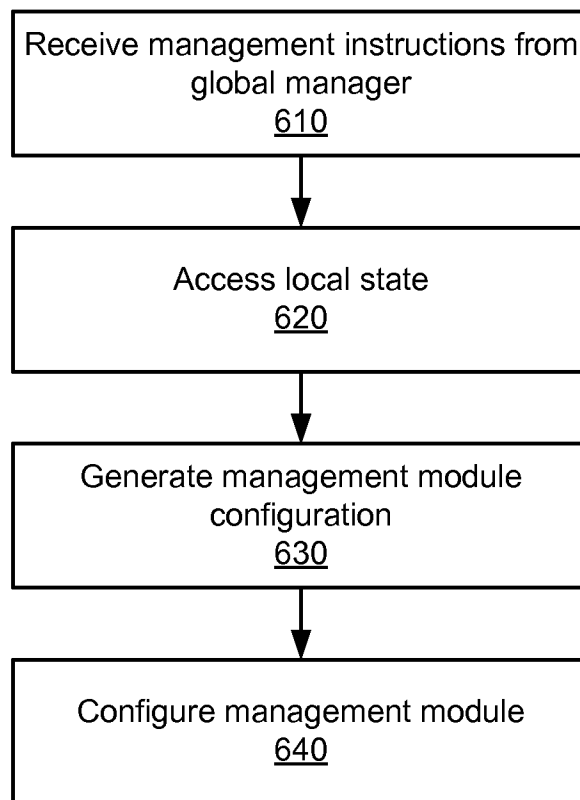


**FIG. 4**

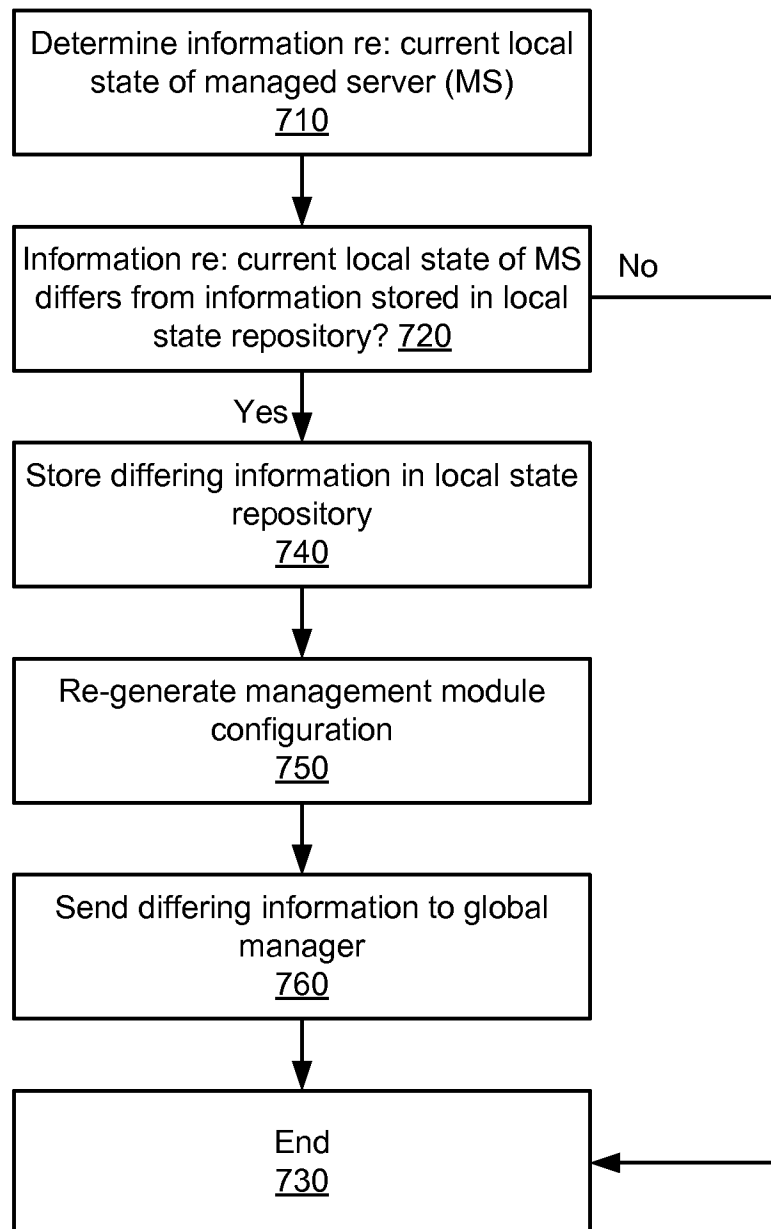
5/11

500**FIG. 5**

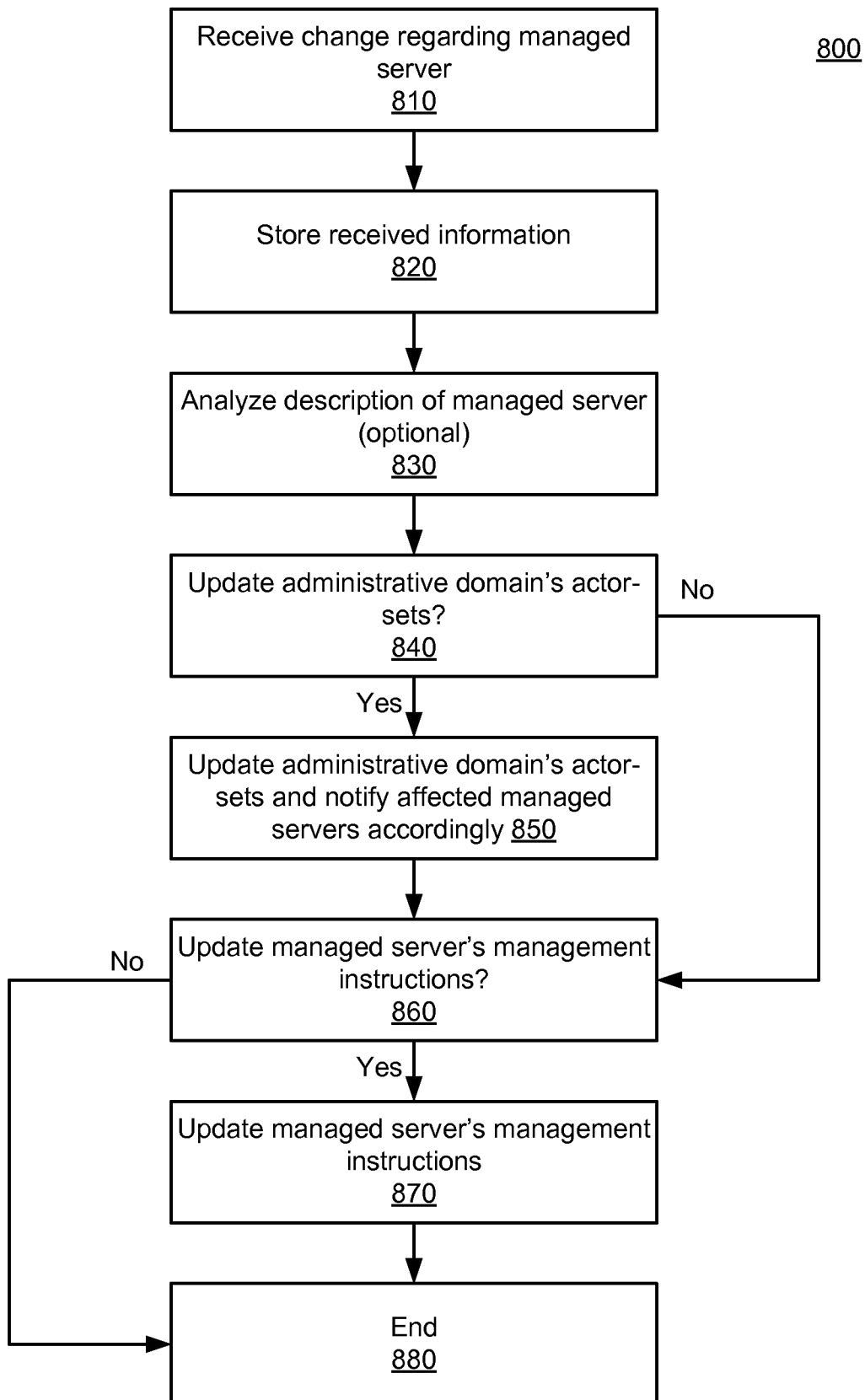
6/11

600**FIG. 6**

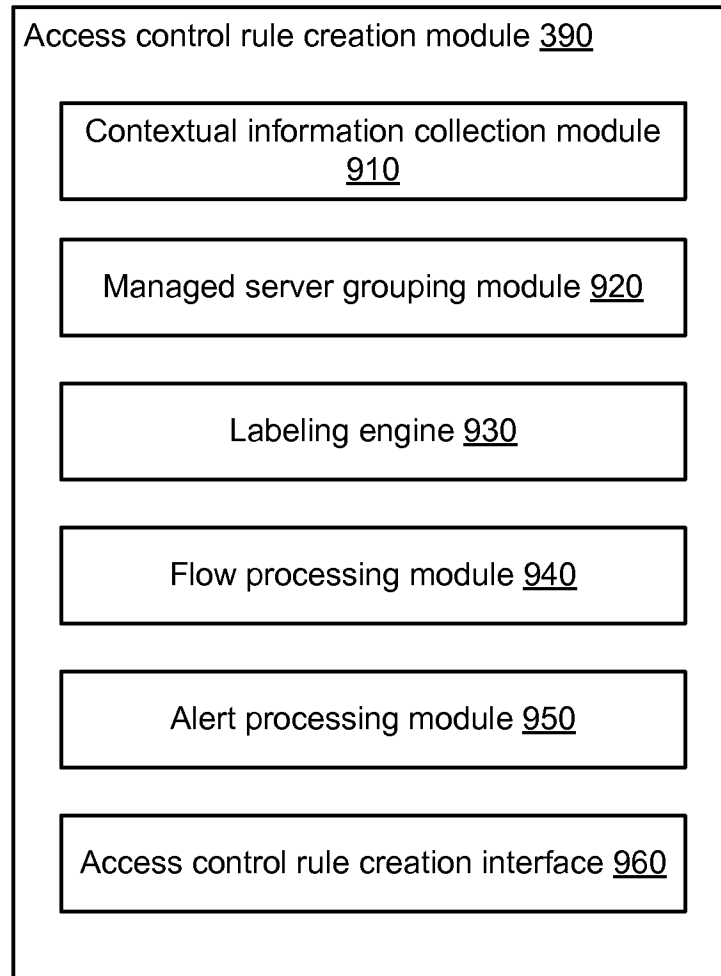
7/11

700**FIG. 7**

8/11

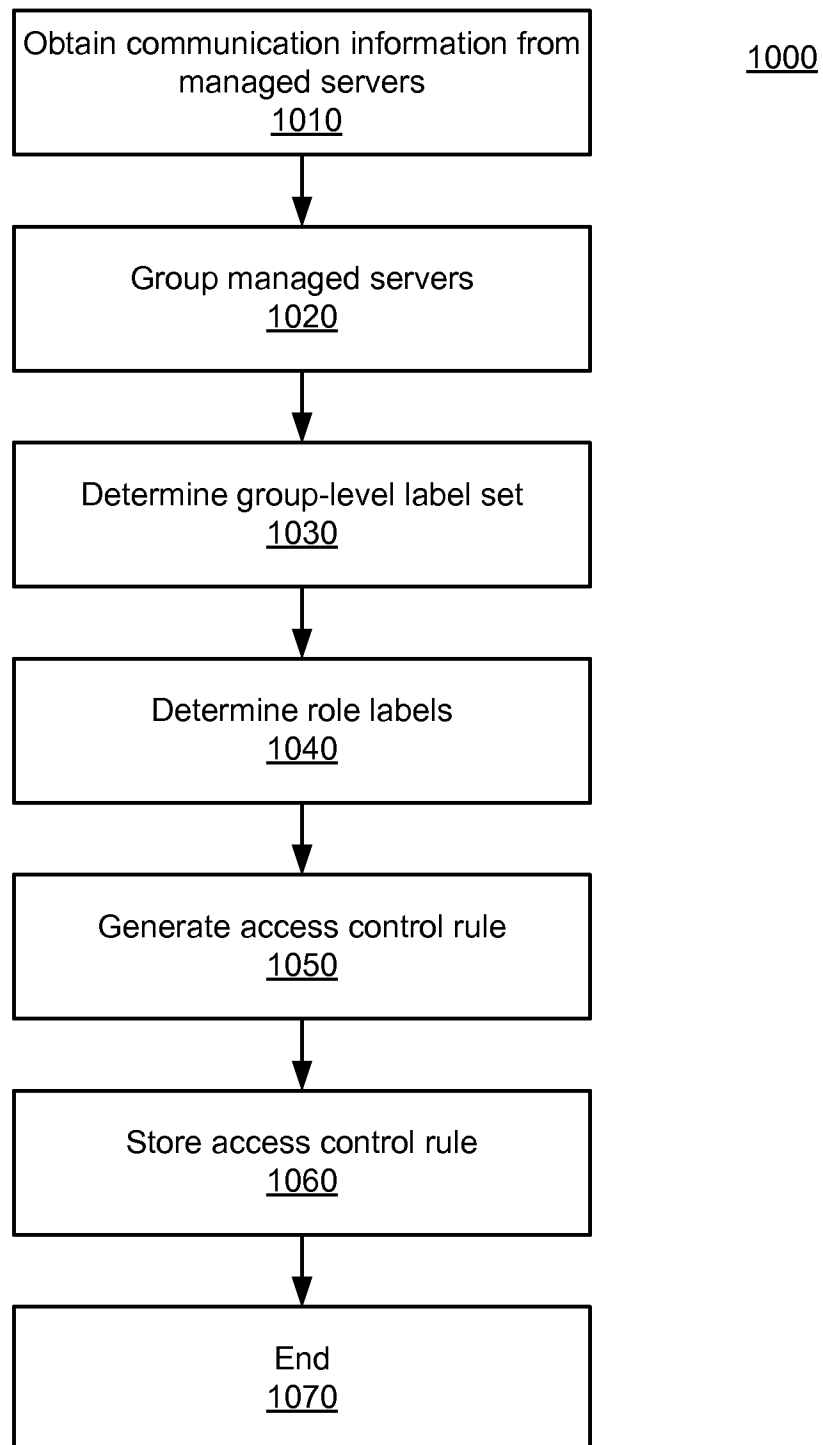
**FIG. 8**

9/11

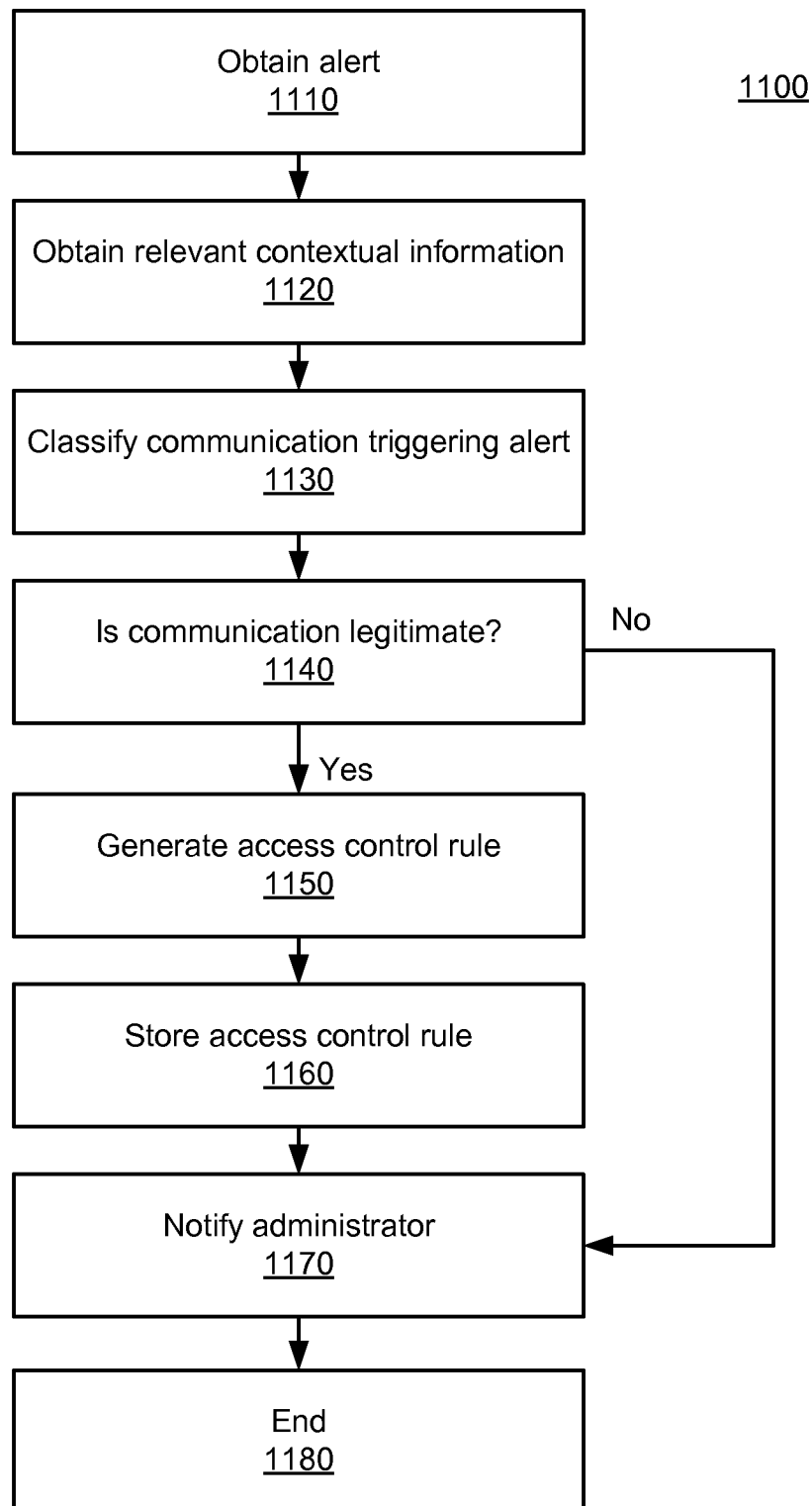
**FIG. 9**



10/11

**FIG. 10**

11/11

**FIG. 11**

# INTERNATIONAL SEARCH REPORT

International Application No. \*

PCT/US 14/63239

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - H04L 29/06 (2015.01)

CPC - H04L 63/20

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

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC (8) - H04L 29/06 (2015.01)

CPC - H04L 63/20

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

USPC - 726/1, 726/27, 709/225, 705/319, 709/224, 718/1, 718/104

CPC - H04L 63/102; G06F 21/6218 (See keywords Below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Thomsoninnovation.com; Patbase; Google Scholar; Google Patents; Gogole.com; Freepatentonline; ProQuest Dialog

Search Terms: Access control, authenticate, authorize, role, privilege, right, server, peer, node, resource, group, cluster, hierarchy, tier, level, label, tag, metadata, generate, policy, rule, criteria, store, storage, previous, past, h

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2011/0209195 A1 (KENNEDY), 25 August 2011 (25.08.2011), entire document, especially Abstract; Para. [0020], [0116], [0120]-[0121], [0123]-[0124], [0130] [0141], [0159]	1-12 and 21
Y	US 2004/0199792 A1 (TAN et al.), 07 October 2004 (07.10.2004), entire document, especially Abstract; Para [0010], [0051], [0144]-[0146]	1-12 and 21
Y	US 2012/0155290 A1 (DANTRESSANGLE et al.), 21 June 2012 (21.06.2012), entire document, especially Abstract; Para [0020], [0064], [0086]-[0089]-	1 and 21
Y	US 2010/0058340 A1 (KAMADA), 04 March 2010 (04.03.2010), entire document, especially Abstract; Para [0027]-[0029], [0032], [0038], [0048]-[0050]	1 and 21
A	US 2009/0165078 A1 (SAMUDRALA et al.), 25 June 2009 (25.06.2009), entire document	1-12 and 21
A	US 6,023,765 A (KUHN), 08 February 2000 (08.02.2000), entire document	1-12 and 21

☐ Further documents are listed in the continuation of Box C.

\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

11 March 2015 (11.03.2015)

Date of mailing of the international search report

26 MAR 2015

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents

P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-3201

Authorized officer:

Lee W. Young

PCT Helpdesk: 571-272-4300

PCT OSP: 571-272-7774

# INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 14/63239

## Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:  
because they relate to subject matter not required to be searched by this Authority, namely:
  
2. ☐ Claims Nos.:  
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
  
3. ☐ Claims Nos.:  
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

## Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:  
This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I: Claims 1-12 and 21 directed to a method and computer program product for determining an access control rule authorizing communication between a plurality of managed servers within an administrative domain.

Group II: Claims 13-20 and 22 directed to a method and system processing alerts from managed servers implementing one or more access control rules.

--- ( See Continuation in Supplemental Box ) ---

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:  
1-12 and 21

### Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

Continuation of:

Box III. Observations where unity of invention is lacking

The inventions listed as Groups I-II do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons:

**Special Technical Features:**

The invention of Group I includes the features of obtaining communication information describing past communication between the plurality of managed servers; identifying a subset of managed servers from the plurality of managed servers by grouping the plurality of managed servers based on the obtained communication information; determining a group-level label set to associate with the subset of managed servers; determining role labels for managed servers in the subset of managed servers, a managed server associated with one role label; generating, based on the group-level label set and the role labels, an access control rule authorizing communication between a first managed server of the subset of managed servers and a second managed server, not required in group II.

The invention of Group II includes the features of obtaining an alert from a first managed server configured to generate the alert in response to past communication with a second managed server and responsive to the first managed server determining that the one or more access control rules do not authorize the past communication between the first managed server and the second managed server; obtaining contextual information including communication information describing the past communication between the first managed server and the second managed server; classifying the past communication as being legitimate or malicious based on the communication information; responsive to classifying the past communication as legitimate, generating an access control rule authorizing the past communication between the first managed server and the second managed server, not required in group I.

**Common Technical Features**

Groups I- II share the technical features of a past communication between the plurality of managed servers; generating an access control rule authorizing communication; and storing the access control rule as part of an administrative domain-wide management policy.

However, the shared technical features does not represent a contribution over prior art as anticipated by US 2011/0209195 A1 (Kennedy), 25 August 2011 (25.08.2011) .

Kennedy teaches a past communication between the plurality of managed servers (Para [0120]- the administrative module 180 tracks not only transmission history of communications but also transmission and user access history); generating an access control rule authorizing communication (Para [0090]- intelligently creates policies and rules by observing system and user behavior); and storing the access control rule as part of an administrative domain-wide management policy (Para [0090], [0111]-[0113]- storage and other such systems to coordinate security, enterprise database 160).

As the common features were known in the art at the time of the invention, this cannot be considered a common technical feature that would otherwise unify the groups.

Therefore, Groups I-II lack unity under PCT Rule 13.