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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 May 2007 (24.05.2007)

PCT

(10) International Publication Number
WO 2007/058946 A2

(51) International Patent Classification:
H04L 12/28 (2006.01)

(21) International Application Number:
PCT/US2006/043800

(22) International Filing Date:
9 November 2006 (09.11.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/272,034 14 November 2005 (14.11.2005) US

(71) Applicant *(for all designated States except US)*: **SOURCEFIRE, INC.** [US/US]; 9770 PATUXENT WOODS DRIVE, Columbia, MD 21046 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VOGEL, William, A.** [US/US]; SOURCEFIRE, INC., 9770 PATUXENT WOODS DRIVE, Columbia, MD (US). **BRUZEK, Dina, L.** [US/US]; SOURCEFIRE, INC., 9770 PATUXENT WOODS DRIVE, Columbia, MD (US).

(74) Agents: **NICHOLSON, Cynthia, K.** et al.; POSZ LAW GROUP, PLC, 12040 SOUTH LAKES DRIVE, SUITE 101, Reston, VA 20191 (US).

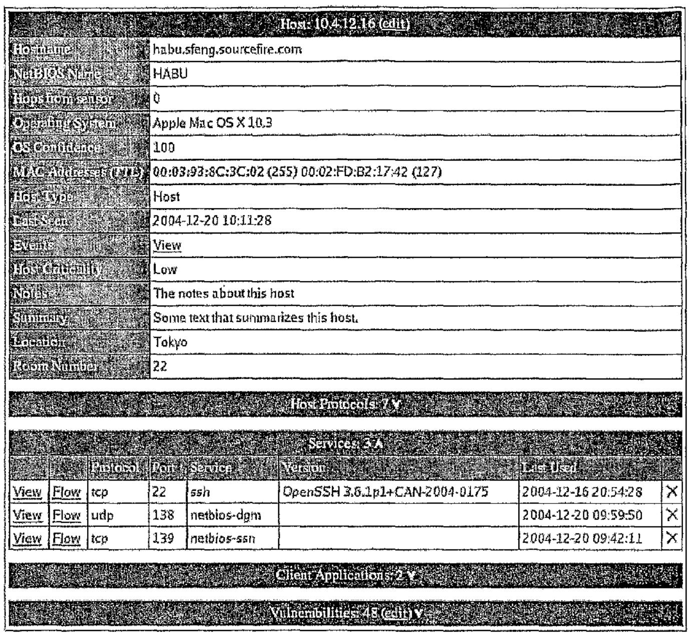
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR,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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR MODIFYING NETWORK MAP ATTRIBUTES



Host: 10.4.12.16 (edit)	
Hostname	habu.sfeng.sourcefire.com
NetBIOS Name	HABU
Hops from sensor	0
Operating System	Apple Mac OS X 10.3
OS Confidence	100
MAC Addresses (TTL)	00:03:93:8C:3C:02 (255) 00:02:FD:B2:17:42 (127)
Host Type	Host
Last Seen	2004-12-20 10:11:28
Events	View
Host Criticality	Low
Notes	The notes about this host
Summary	Some text that summarizes this host.
Location	Tokyo
Room Number	22

Host Protocols: 7 ▼

Services: 3 ▲

		Protocol	Port	Service	Version	Last Used	
View	Flow	tcp	22	ssh	OpenSSH 3.6.1p1+CAN-2004-0175	2004-12-16 20:54:28	X
View	Flow	udp	138	netbios-dgm		2004-12-20 09:59:50	X
View	Flow	tcp	139	netbios-ssn		2004-12-20 09:42:11	X

Client Applications: 2 ▼

Vulnerabilities: 48 (edit) ▼

(57) **Abstract:** The disclosed systems and methods provide a user interface for modifying host configuration data that has been automatically and passively determined and for adding or modifying other parameters associated with a host. A host data table can store various parameters descriptive of a host including the applicability of specific vulnerabilities. If it is determined that one or more hosts should not be identified as associated with a specific vulnerability, a graphical user interface can be used to modify the vulnerability parameter.

SYSTEMS AND METHODS FOR MODIFYING NETWORK MAP ATTRIBUTES

Field of the Invention

[0001] Embodiments of the present invention relate to systems and methods for determining the characteristics of a computer network. More particularly, embodiments of the present invention relate to systems and methods for automatically and passively determining a host configuration of a computer network and providing a user interface for modifying the host configuration data.

Background

[0002] Computers and computer networks connecting such computers are vital components of modern society. Unfortunately, such computer networks are susceptible to attacks from internal and external hostile sources. Intrusion detection systems (IDSs) are used to prevent such attacks. Conventional IDSs operate by analyzing network traffic in terms of the traffic itself. They do not, however, consider the end points of that traffic. End points refer to the originators and recipients of message traffic. Such end points include, for example, clients and the servers. Analysis of end points can provide contextual information about the network such as host addresses and services. By not considering these end points, a vital piece of contextual information about the network is missed. Consequently, a substantial need exists for technologies that provide information about the end points of computer network traffic.

[0003] IDSs are less effective than they could be because they do not have contextual information about the computer network they are monitoring. For example, without contextual information, IDSs are susceptible to a computer network attack known as evasion. Evasion occurs when an attacker uses network endpoint information that the IDS does not have, to evade detection by the IDS. A known method of evasion is insertion. Insertion can be used in networks having a routing infrastructure that handles packets of different sizes. For example, a routing link (router or some other device) may be attached to a network that supports a 1500 byte maximum size on one side of the device and 500 bytes on the other. If someone was trying to talk to a host on the other side of the device, the maximum packet size they could send is 500 bytes. This maximum is called the "Path MTU" (Maximum Transfer Unit). If an attacker knows this, they can transmit a large packet between two properly sized packets and get the IDS to accept the oversized packet, giving the IDS a bad model of the data that is actually arriving at the host.

[0004] Not only does the lack of contextual information make the IDS more susceptible to attack, but it also makes the IDS less efficient. One such inefficiency is that, without contextual information, the IDS may not be able to discern whether or not an attack can cause harm. Attacks can be directed to a particular service running on a target host. Without information about the services running on the target host, an IDS could mistakenly detect an attack even if that host is not running the targeted service. That is, the IDS would cause an alarm even though the attack would be harmless. Such an event is called a false positive. Large numbers of false positives can make it more difficult and expensive to locate genuine attacks that can harm a host on the network.

[0005] Some conventional techniques for providing contextual information to IDSs are known. One such technique is for a human to audit each host manually and gather all desired contextual information. This manual method has a number of disadvantages including that it is time consuming, prone to error, and makes maintenance more difficult. One reason for these drawbacks is that networks are dynamic in nature. Host computers in computer networks are added, removed, and reconfigured. If these changes are not meticulously documented, each computer on the network must be revisited periodically to insure that the contextual information is up to date.

[0006] Another conventional technique for providing contextual information to an IDS is an automatic discovery system. Conventional automatic discovery systems are active scanning systems that actively probe end hosts on a computer network and perform stimulus response tests on them to find and record vulnerabilities that exist on end hosts. Though not manual, active scanning systems also suffer from several problems. One problem is that active scanning can be destructive to the network. In testing for vulnerabilities, they can cause both routers and servers to malfunction or stop functioning. Another problem is that they may not provide information useful to an IDS because in many instances a one-to-one mapping does not exist between the information an active scanner provides and the information an IDS can use. Another problem is that active scanners only provide a snapshot of the network at the time when the scan is performed. This snapshot is problematic because a host may run a vulnerable service transiently. In such a case, the active scanning may be performed at a time when the vulnerable service is not running. As a result, the active scan would not cause an alarm despite the transient nature of the vulnerability.

[0007] While it is advantageous to automatically and passively determine a host configuration, there may be occasions when a passively determined map does not incorporate information that would be useful. For example, a certain version of a service may be known to have certain vulnerabilities while another version of the same service may not be vulnerable. In some cases, a patch may have been applied to a service running on a host thereby eliminating a known vulnerability. In cases such as these, it is advantageous to provide an interface by which a user can modify the data stored in the network map. It is a further advantage to allow a user to conveniently annotate network map data with a graphical user interface.

[0008] In view of the foregoing, it can be appreciated that a substantial need exists for systems and methods that can advantageously provide a user interface for modifying an automatically and passively determined host configuration of a computer network.

Summary of the Invention

[0009] One embodiment of the invention includes a method for assigning a vulnerability parameter to a device on a network by defining a vulnerability parameter for an operating system or service, storing the vulnerability parameter in a host map associated with a network device, providing a graphical user interface for viewing and modifying the vulnerability parameter, and storing the modified vulnerability parameter in the host map associated with the service or operating system.

[0010] Another embodiment of the invention includes a method for automatically and passively determining the characteristics of a network by reading one or more packets transmitted on the network, identifying a network device on the network using the one or more packets, recording an identity of the network device in a host map record associated with the network device, providing a graphical user interface for receiving a parameter from a user, and recording the user parameter in the host map record associated with the network device.

[0011] Another embodiment of the invention is a system for assigning a vulnerability parameter to a device on a network including a computer-readable medium for storing a vulnerability parameter for an operating system or service in a host map associated with a network device, a display for displaying a graphical user interface for viewing and modifying the vulnerability parameter assigned to the service or operating system, and a computer-readable medium for storing the modified vulnerability parameter in the host map associated with the service or operating system.

[0012] Another embodiment of the invention is a system for automatically and passively determining the characteristics of a network including a receiver for receiving one or more packets transmitted on the network and identifying a network device on the network using the one or more packets, a computer-readable medium for storing an identity of the network device in a host map, a display for displaying a graphical user interface for receiving a parameter from a user, and a computer-readable medium for storing the user parameter in the host map associated with the identified network device.

Brief Description of the Drawings

[0013] Figure 1 illustrates an exemplary method for automatically and passively determining the characteristics of a network.

[0014] Figure 2 illustrates an exemplary data structure for storing network device information or host information.

[0015] Figure 3 illustrates an exemplary user interface for displaying user-defined host attributes.

[0016] Figure 4 illustrates an exemplary user interface for adding and naming a new attribute and selecting its type.

[0017] Figure 5 illustrates an exemplary user interface for defining an integer attribute with a range.

[0018] Figure 6 illustrates an exemplary user interface for adding or selecting list items.

[0019] Figure 7 illustrates an exemplary user interface for modifying an attribute in the attribute list.

[0020] Figure 8 illustrates an exemplary interface for selecting an attribute from a list of user-defined attributes.

[0021] Figure 9 illustrates an exemplary user interface including collapsed subsections.

[0022] Figure 10 illustrates an exemplary interface for editing attributes in a table view.

[0023] Figure 11 illustrates an exemplary vulnerability editor.

[0024] Figure 12 illustrates an exemplary user interface for displaying "Invalid" or "Not Applicable" vulnerabilities.

[0025] Figure 13 illustrates an exemplary interface for searching a criticality field.

[0026] Figure 14 illustrates an exemplary interface for setting a criticality field.

[0027] Figure 15 illustrates an exemplary interface for selecting the criticality level for a host.

[0028] Figure 16 illustrates an exemplary interface for setting a criticality field.

[0029] Figure 17 illustrates an exemplary text or notes field.

Detailed Description of the Invention

[0030] Embodiments of systems and methods for modifying a passively determined network characteristic database are described in this detailed description of the invention. In this detailed description, for purposes of explanation, numerous specific details are set forth to provide a thorough understanding of embodiments of the present invention. One skilled in the art can appreciate, however, that embodiments of the present invention may be practiced without these specific details. In other instances, structures and devices are shown in block diagram form. Furthermore, one skilled in the art can readily appreciate that the specific sequences in which methods are presented and performed are illustrative and it is contemplated that the sequences can be varied and still remain within the spirit and scope of embodiments of the present invention.

[0031] The present invention is applicable to IDSs, such as those described in the co-pending U.S. Patent Application Serial No. 10/793,887, filed March 8, 2004, titled "Methods and Systems for Intrusion Detection," by Marc A. Norton and Daniel J. Roelker, which is herein incorporated by reference in its entirety. The present invention is also applicable to analysis of vulnerabilities, as described in the co-pending U.S. Patent Application Serial No. 10/843,353, filed May 12, 2004, titled "Systems and Methods for Determining Characteristics of a Network and Analyzing Vulnerabilities," by Martin Roesch, William Andrew Vogel, III, and Matt Watchinski, which is herein incorporated by reference in its entirety.

[0032] Embodiments of the present invention can be applied to passively determined network maps or characteristic databases. These systems are passive because they examine packets moving across a network; they do not perform active scans. They are automatic because they require little or no human intervention. Such passive systems operate by performing functions including: (1) identifying each network device on a network, (2) identifying operating system and services running on each network device, (3) recording, in real-time, any changes occurring on the network, and (4) gathering this information in a format that can be used by a network reporting mechanism. Exemplary network reporting mechanisms include IDSs and network management systems (NMSs).

[0033] Network reporting mechanisms can examine packets moving across a network in real-time, for characteristic information about the network. One such type of characteristic information is information related to a network device, or host, on the network. One skilled in the art can appreciate that a network device is any device with a network connection. Network devices include but are not limited to computers, printers, switches, game machines, and routers.

[0034] In response to network traffic, the identity of the network device is recorded. The identity is stored as a data structure in a file or database, for example. If a packet identifies a network device that has previously been recorded, the current information and the previous information are compared and any changes are recorded. If no changes have been found, no new information is recorded. In either case, the next packet is read.

[0035] Figure 1 is a flowchart showing an exemplary method 100 for automatically and passively determining the characteristics of a network in accordance with an embodiment of the present invention.

[0036] In step 110 of method 100, a packet transmitted on a network is read.

[0037] In step 120, a network device is identified using the packet. A network device includes but is not limited to a computer, a printer, and a router. One skilled in the art can appreciate that a network device can also be referred to as a host.

[0038] In step 130, the identity of the network device is recorded. The identity is stored as a data structure in a file or database, for example. If a packet identifies a network device that has previously been recorded, the current information and the previous information are compared and changes can be recorded. If no changes have been found, no new information is recorded. In either case, method 100 returns to step 110 to read the next packet.

[0039] Figure 2 is an exemplary data structure used to store network device information, or host information. This data structure is the host representative data structure. As non-limiting examples, the host information can includes the initiator Internet protocol (IP) address, a list of media access control (MAC) addresses with a time-to-live (TTL) parameter for each MAC address, a list of operating systems, a list of network protocols, a list of transport protocols, a list of transmission control protocol (TCP) service data structures, a list of user datagram protocol (UDP) service data structures, a virtual local area network (VLAN) tag, and a last seen time. The IP address, MAC address and TTL parameter of at least one network device on the network are typically included in each packet transmitted on the network. As a result, these pieces of host information are obtained by directly parsing the network and transport protocol fields of each packet.

Vulnerability Analysis

[0040] In some network monitoring systems, vulnerabilities are assigned to hosts discovered on a network. Vulnerabilities are known methods of maliciously gaining access to a host or host service, or maliciously attacking a host or host service. The vulnerabilities assigned to a host are derived from various sources and this information can be stored in the data structures discussed above for storing network device information. IDSs for example, can maintain vulnerability lists.

[0041] A list of potential vulnerabilities can be stored in a vulnerabilities database (VDB). When a host or host service is identified, one or more vulnerabilities from the VDB is mapped in real-time to the host or host service. These vulnerabilities can then be displayed in a graphical user interface, linked to the particular host or service. An administrator can use this information in connection with the patching specific systems or groups of systems.

[0042] In some cases, a discovered host can be actively scanned in order to refine the list of vulnerabilities for that host from a list of all possible vulnerabilities to a smaller set of core vulnerabilities. This elimination of vulnerabilities along with the lowering of the priority of vulnerabilities that have either already been patched on the target system or vulnerabilities that are not currently present due to the configuration of the service, allows the administrator to target efforts to resolve these vulnerabilities to identified problem areas.

[0043] The present invention permits the creation and modification of predefined and user-defined host attributes with predefined or user-defined values. It also provides the ability for the user to adjust these attribute values for any host manually. In some embodiments, the host attributes can be modified through a graphical user interface (GUI) provided through an application program interface (API). Through this interface, the user can have the ability to modify the network map and/or database through the graphical user interface. As non-limiting examples, the user can be provided with functions for (a) deletion of hosts and network subnets from the network map, (b) deletion of services from the network map, (c) marking of vulnerabilities as "not applicable," (d) modification of a criticality field for host entries, and (e) text annotation of hosts.

Selective Marking of Vulnerability Mappings as "Not Applicable"

[0044] Users can mark vulnerabilities as "Not Applicable" for a particular host. Vulnerabilities that are "Not Applicable" can be tagged with a database field in the network device database indicating that they are invalid (or not applicable) for the specific host or service. As a result, the "Not Applicable" vulnerabilities can be displayed in a separate list in the host profile and an impact flag can be set based on these vulnerabilities. After a vulnerability has been marked as "Not Applicable," the user can reactivate the vulnerability. Reactivation of a vulnerability unsets the database flag, causes the impact flag to use the vulnerability again, and causes display of the vulnerability in the "Valid" or "Applicable" vulnerabilities section in the user interface. An example user interface showing "Invalid" or "Not Applicable" vulnerabilities is shown in Figure 12.

[0045] In some embodiments, a vulnerability marked as "Not Applicable" can lose status as "Not Applicable" on the occurrence of one or more predefined events. For example, if a change to an operating system or service on an entity occurs that causes a new vulnerability lookup to be performed, the vulnerability can be remapped to the entity as a "Valid" vulnerability, if applicable. If an operating system change event occurs, the previous vulnerability list can also be deleted and a new vulnerability lookup can be performed. If a vulnerability exists for an operating system and is marked as "Invalid" before an operating system change event, the vulnerability can be remapped as a "Valid" vulnerability.

Multiple Vulnerability Modification

[0046] In some embodiments, vulnerabilities can be marked as "Not Applicable" for several hosts with one user action. In such embodiments, multiple hosts or services mapped with a particular vulnerability are searched and identified and all of them are marked as "Not Applicable" without requiring the user to manually select each host or service.

User-Defined Attributes

[0047] Some embodiments of the invention can include an interface for creating one or more attributes in the host profile and assigning values or data types to the attribute. For example, attributes can include (a) an operational criticality attribute with values such as high, medium, or low, (b) a location attribute for storing exemplary values such as "Rockville," "Santa Clara," "London"; (c) an room number attribute for storing integer values type in a predefined range such as 0 to 600.

Attribute Types

One or more of the following types of attributes may be edited by a user through a graphical user interface.

Text Attribute Type

[0048] This attribute type is a simple text field that can be appended to or included in the host profile. Removing special characters that may cause security issues from the attribute text value can be handled automatically. An exemplary text or notes field is shown in Figure 17.

Integer Range Type

[0049] This attribute type is defined as an integer between a user defined minimum and maximum. If the user attempts to assign an integer to this attribute that falls outside the defined range, an error can be returned. If the attribute definition is modified, any host attribute setting that falls outside the defined range can be deleted.

Value List Type

[0050] This attribute type is defined as a list of string values. At least one value can be defined per list. Each entry can contain an ID that can be used to identify the attribute value when it is assigned to a host. If the attribute is modified and an item is deleted, any host attribute setting with this item value can also be deleted. For example, if a "Location" list attribute exists, and a system is assigned to the "Rockville" location, if the "Rockville" location is deleted, the Location attribute is deleted from all hosts with the "Rockville" value.

IP Assigned Value List Type

[0051] This attribute type is similar to the Value List Type, except that each list item can be associated with a list of network definitions (IP address/netmask pair in CIDR notation), and the value of the attribute for a host can be set automatically based on its IP address. If the value for a specific list item is modified, other hosts having that value set can be updated automatically.

[0052] The network definitions can be evaluated from most-specific to least-specific. For example, if the networks 10.4.0.0/16 and 10.0.0.0/8 are specified for different levels, the host 10.4.12.68 can be assigned the value matched with the 10.4.0.0/16 network. If the IP ranges for list items are modified, then the attribute value settings for the hosts can be automatically re-evaluated.

Host Criticality

[0053] A server criticality field can be provided for hosts in the network map. This field can be added in both the database and memory map representations of hosts and can be a searchable field. An exemplary interface for searching this attribute is shown in Figure 13.

[0054] In one exemplary embodiment, the criticality category can contain four levels: none, low, medium, and high. The "none" level can be defined as the default level. When new hosts are added to the network map they can automatically be assigned this level. Users can change the level for individual hosts or apply a level change to selected hosts from the search results screen and/or the host view screen. Exemplary interfaces for setting the criticality field are shown in Figures 15 and 17. As discussed in more detail below, the criticality field can also be used in a rule associated with one or more policies and responses.

[0055] From the host map view, the criticality of a specific host may be selected by a user. According to the exemplary interface shown in Figure 15, if the user clicks on a link, a pop-up can appear to allow the user to select the criticality level for the specific host.

Network Monitoring System Integration

[0056] Attribute definitions can contain Universal Unique Identifiers (UUIDs) to allow high availability peers and managed sensors to differentiate between locally and remotely generated attributes. As used herein, a high availability peer is a second system storing a second copy of a network map. In some embodiments, the network map stored for high availability can be synchronized between multiple peers hosting the map. Hosts assigned a value on one network monitoring system can be automatically assigned that value on a second high availability network monitoring system. In some embodiments, attributes that have been propagated to a second network entity can be modified on the second entity and propagated back to the first entity. Any of attributes and/or validity and applicability parameters described herein can be made available through high availability peers.

[0057] Users can be given the ability to define generic attributes that can be applied to all host profiles. The attributes can be usable in host searches and in connection with policies and responses. Features such as, but not limited to, adding, deleting and modifying attribute definitions, and setting and deleting attribute values for specific hosts can be performed through an API.

Policy and Response

[0058] When attributes are created, these values can be added to policy and response rules and search constraints. For the value list attribute type, users can be allowed to use the "is" and "is not" operators. For the integer range attribute type, users can be allowed to use mathematical operators such as =, <, and >. For text attributes, regular expression searches can be performed using operators such as "contains", "starts with", and "ends with."

[0059] In further embodiments, an interface can be provided so that a user can (a) assign attributes and values to hosts in the host profile, (b) assign attributes to one or more hosts in a host table view, (c) automatically assign and automatically set attribute values to a host based on an IP address and when the host is detected, (d) set predefined host attributes for use in a policy and response rule as a host qualifier, and (e) manage user defined host attributes.

Graphical User Interface

[0060] The graphical user interface can provide the user with a vulnerability editor to mark a specific vulnerability as "not applicable" for one or more hosts. In some embodiments, the interface can include a search screen that includes "vulnerability ID" as a searchable field to facilitate location of hosts with a selected vulnerability. Some embodiments can include a user interface to support user-defined host attributes. Through the interface, the user can perform one or more of the following functions: (a) creating and modifying attribute definitions, (b) viewing and modifying attribute settings for specific hosts, and (c) organizing host displays based on attribute values.

Existing Attribute List

[0061] The attribute list screen displays user-defined attributes. Initially, the list may be empty. Activation of a "create" link can display a separate screen for creating new attributes. For existing attributes, the table can display a name, type, and automatically assign a setting for attributes. The interface can include two links in each row for modifying and deleting specific attributes: an "edit" link can display an interface for editing entries and a "delete" link can delete selected items and refresh the table display. One such exemplary interface is shown in Figure 3.

Add New Attribute

[0062] An add new attribute screen allows the user to name a new attribute, choose its type, and configure further settings based upon the type. As discussed above, exemplary types of attributes can include: text, integer, and list. In some embodiments, a limited set of the graphical user interface components for configuring the currently selected type are displayed. One such exemplary interface is shown in Figure 4.

[0063] The integer range type interface can display two new rows containing the minimum and maximum input fields for the integer range. One such exemplary interface is shown in Figure 5.

[0064] The list type can display two or more lists. One such exemplary interface is shown in Figure 6. The first list is the list of values for the attribute. When one of these values is selected, the second list can display the IP ranges that are automatically assigned that value. Values can be added to either list using input fields. Selected items can be deleted using the "X" in the top-right of each list area. When the user is finished configuring the new attribute, the "Save Attribute" button can be used to add the attribute. The user can return to the attribute list later by activating a "Return to Attribute List" link or button.

Modify Existing Attribute

[0065] When the user clicks on the modify link for a specific attribute in the attribute list, an interface for modifying an existing attribute can be presented. One such exemplary interface is shown in Figure 7. In some embodiments, a user is not permitted to change the type for an existing attribute. In such embodiments, the type is displayed as a label instead of a modifiable combination box. In some embodiments, controls employed in the new attribute interface can be used in the modify existing attribute interface.

Network Map View of Attributes

[0066] The attribute view can allow the user to choose an attribute from a list of user-defined attributes. The values for the selected attribute can then be displayed as the top-level elements in the network map tree. When the user expands one of these values, a list of host IPs that have that attribute value set can be displayed as child elements. Selection of one of these hosts can cause display the host profile view in the right-hand panel. One such exemplary interface is shown in Figure 8.

Attributes in the Host Profile

[0067] Host profile data can be viewed in any one of several graphical user interfaces. In some embodiments, editing of host attributes and vulnerabilities can be performed in separate popup windows. Attributes that have values set can also be displayed in a read-only host profile.

Collapsed Subsections

[0068] In some embodiments, displayed subsections, such as Services and Vulnerabilities can be expanded or collapsed. By default, they can be collapsed which can make the host profile less overwhelming and reduce its vertical size. The collapsed subsections can display a total element count in the title bar to inform user of the number of sub-elements without requiring expansion of each section. One such exemplary interface is shown in Figure 9.

Attribute Editor

[0069] An attribute editor can be used to edit attributes in the table view. The editor can be a popup window that allows the user to select an available attribute and set its value for the selected host. The popup can include of a presentation of available attributes. Once an attribute is selected, an appropriate edit field for that attribute can be displayed. As non-limiting examples, the fields can include a text-area for text attributes, a text-field for range attributes, and a box for list attributes. One such exemplary interface is shown in Figure 10.

Vulnerabilities Editor

[0070] The vulnerability editor can allow users to easily designate the vulnerabilities that are, and are not, applicable to the current host. In one embodiment, the window can consist of two select lists, side by side, with arrow buttons between them. To transfer vulnerabilities from one list to the other, the user can select the desired items and click one of the arrow buttons. When the user is finished arranging the lists, the "Save" button can be used to apply the changes. One such exemplary interface is shown in Figure 11.

Attribute Table View

[0071] An attribute table view can be used for searching, displaying, and reporting of attribute values and the hosts associated with those values. These attributes can be displayed in the host table view or in a separate table view. This table view can display columns for the attribute name, the IP address of hosts with attribute settings, and the value of those attributes.

[0072] A corresponding attribute search page can be provided for searching the attribute table view based on attribute name, value, and/or host IP address. Link buttons from the attribute table view to other table views can also be used for locating other information associated with a set of selected attributes.

[0073] In some embodiments, a "Set Attributes" button can be included in the interface. Selection of this button can cause display a popup window that allows the user to select an available attribute and set its value for hosts identified in the table view. The popup can include a combo-box of available attributes, and once an attribute is selected, an appropriate edit field for that attribute can be displayed such as a text-area for text attributes, a text-field for range attributes, or a combo-box for list attributes.

[0074] In accordance with an embodiment of the present invention, instructions adapted to be executed by a processor to perform a method are stored on a computer readable medium. The computer-readable medium can be a device that stores digital information. For example, a computer-readable medium includes a read-only memory (e.g., a Compact Disc-ROM ("CD-ROM") as is known in the art for storing software. The computer-readable medium can be accessed by a processor suitable for executing instructions adapted to be executed. The terms "instructions configured to be executed" and "instructions to be executed" are meant to encompass any instructions that are ready to be executed in their present form (e.g., machine code) by a processor, or require further manipulation (e.g., compilation, decryption, or provided with an access code, etc.) to be ready to be executed by a processor.

[0075] Embodiments of the present invention relate to data communications via one or more networks. The data communications can be carried by one or more communications channels of the one or more networks. A network can include wired communication links (e.g., coaxial cable, copper wires, optical fibers, a combination thereof, and so on), wireless communication links (e.g., satellite communication links, terrestrial wireless communication links, satellite-to-terrestrial communication links, a combination thereof, and so on), or a combination thereof. A communications link can include one or more communications channels, where a communications channel carries communications.

[0076] A system and method in accordance with an embodiment of the present invention disclosed herein can advantageously improve existing intrusion detection systems or real-time network reporting mechanisms by giving them contextual information about a computer network. Such a system and method is particularly advantageous in comparison to manual methods in that its information is updated automatically. The ability of the present invention to discover the operating systems of both servers and clients is an important advantage over conventional network detection systems. It is advantageous in comparison to active scanning systems in that it is not destructive to the network, it can provide relevant information to an IDS and its information is always up to date. It can be used to provide information to enhance intrusion detection systems or to provide continuous real-time reports of the status of the network. It can discover the operating systems of both servers and clients.

[0077] In the foregoing detailed description, systems and methods in accordance with embodiments of the present invention have been described with reference to specific exemplary embodiments. Accordingly, the present specification and figures are to be regarded as illustrative rather than restrictive. The scope of the invention is to be further understood by the claims, and by their equivalents.

Claims

1. A method for assigning a vulnerability parameter to a device on a network, comprising:

defining a vulnerability parameter for an operating system or service;

storing the vulnerability parameter in a host map associated with a network device;

providing a graphical user interface for viewing and modifying the vulnerability parameter; and

storing the modified vulnerability parameter in the host map associated with the service or operating system.

2. The method of claim 1, wherein the vulnerability parameter is invalid or valid.

3. The method of claim 1, wherein the graphical user interface is provided through an application program interface.

4. The method of claim 1, wherein the graphical user interface provides a function for setting a vulnerability parameter for a plurality of selected hosts or services.

5. The method of claim 1, wherein the vulnerability parameter is associated with a universal unique identifier and is synchronized to a high availability peer.

6. The method of claim 1, wherein the graphical user interface provides a function for deleting a host from a network map.

7. A method for automatically and passively determining the characteristics of a network, comprising:

reading one or more packets transmitted on the network;

identifying a network device on the network using the one or more packets;

recording an identity of the network device in a host map record associated with the network device;

providing a graphical user interface for receiving a parameter from a user; and

recording the user parameter in the host map record associated with the network device.

8. The method of claim 7, wherein the graphical user interface is provided through an application program interface.

9. The method of claim 7, wherein the parameter is an operational criticality attribute.

10. The method of claim 7, wherein the parameter is a location attribute.

11. The method of claim 7, wherein the parameter is an integer in a user-defined range.

12. The method of claim 7, wherein the parameter is a list of string values.

13. The method of claim 7, wherein the parameter is associated with a universal unique identifier and is synchronized to a high availability peer.

14. A system for assigning a vulnerability parameter to a device on a network, comprising:

a computer-readable medium for storing a vulnerability parameter for an operating system or service in a host map associated with a network device;

a display for displaying a graphical user interface for viewing and modifying the vulnerability parameter assigned to the service or operating system; and

a computer-readable medium for storing the modified vulnerability parameter in the host map associated with the service or operating system.

15. The system of claim 14, wherein the vulnerability parameter is invalid or valid.

16. The system of claim 14, wherein the vulnerability parameter is associated with a universal unique identifier and is synchronized to a high availability peer.

17. A system for automatically and passively determining the characteristics of a network, comprising:

a receiver for receiving one or more packets transmitted on the network and identifying a network device on the network using the one or more packets;

a computer-readable medium for storing an identity of the network device in a host map;

a display for displaying a graphical user interface for receiving a parameter from a user; and a computer-readable medium for storing the user parameter in the host map associated with the identified network device.

18. The system of claim 17, wherein the parameter is associated with a universal unique identifier and is synchronized to a high availability peer.

19. The system of claim 17, wherein the graphical user interface is provided through an application program interface.

20. The system of claim 17, wherein the parameter is an operational criticality attribute.

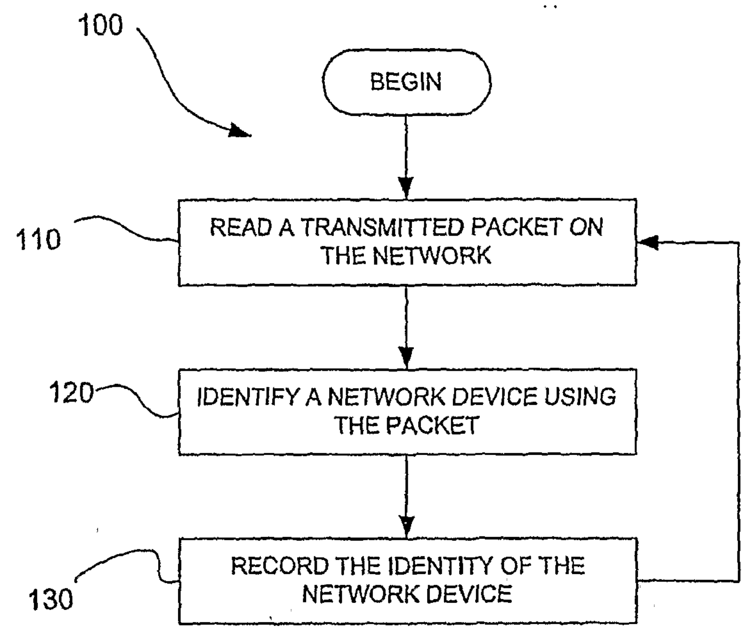
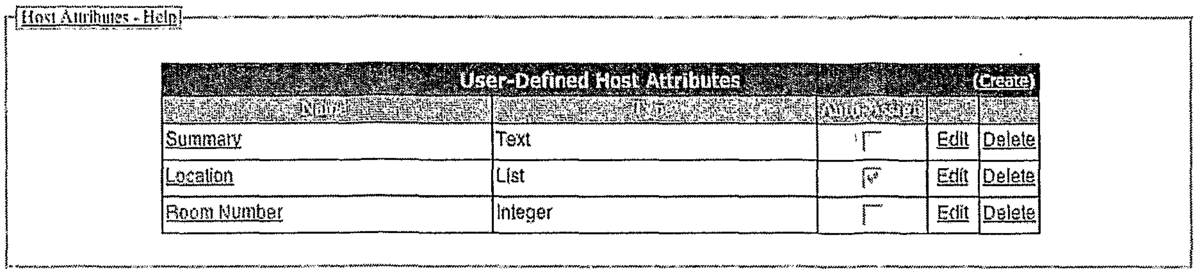

100
BEGIN
110
READ A TRANSMITTED PACKET ON THE NETWORK
120
IDENTIFY A NETWORK DEVICE USING THE PACKET
130
RECORD THE IDENTITY OF THE NETWORK DEVICE

FIG. 1

Host Representative Data Structure

IP address
List of MAC addresses with Time to Live
List of Operating Systems
List of Network Protocols
List of Transport Protocols
List of TCP Service Data Structures
List of UDP Service Data Structures
VLAN tag
Last seen time

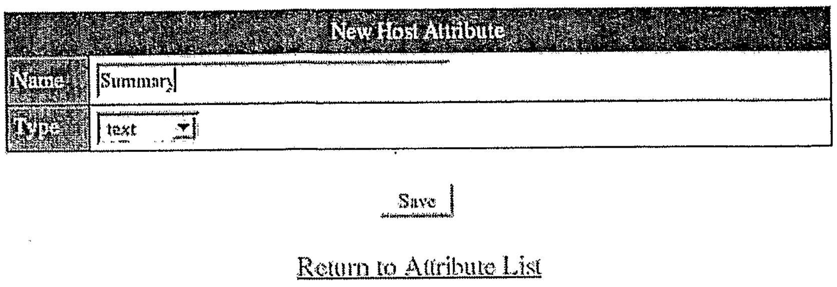
FIG. 2



Host Attributes - Help

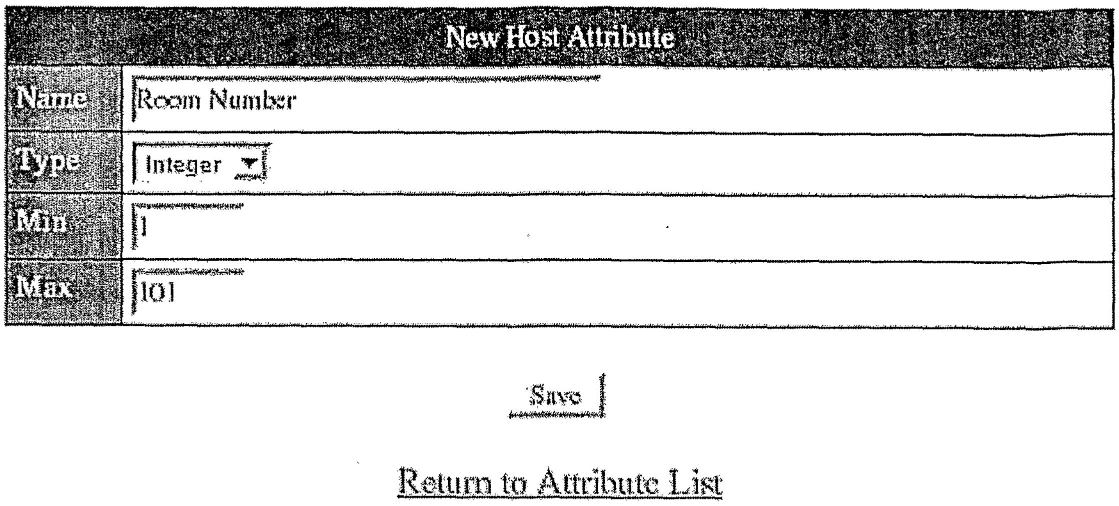
User-Defined Host Attributes				(Create)
Name	Type	Auto Assign		
Summary	Text	☐	Edit	Delete
Location	List	✓	Edit	Delete
Room Number	Integer	☐	Edit	Delete

FIG. 3



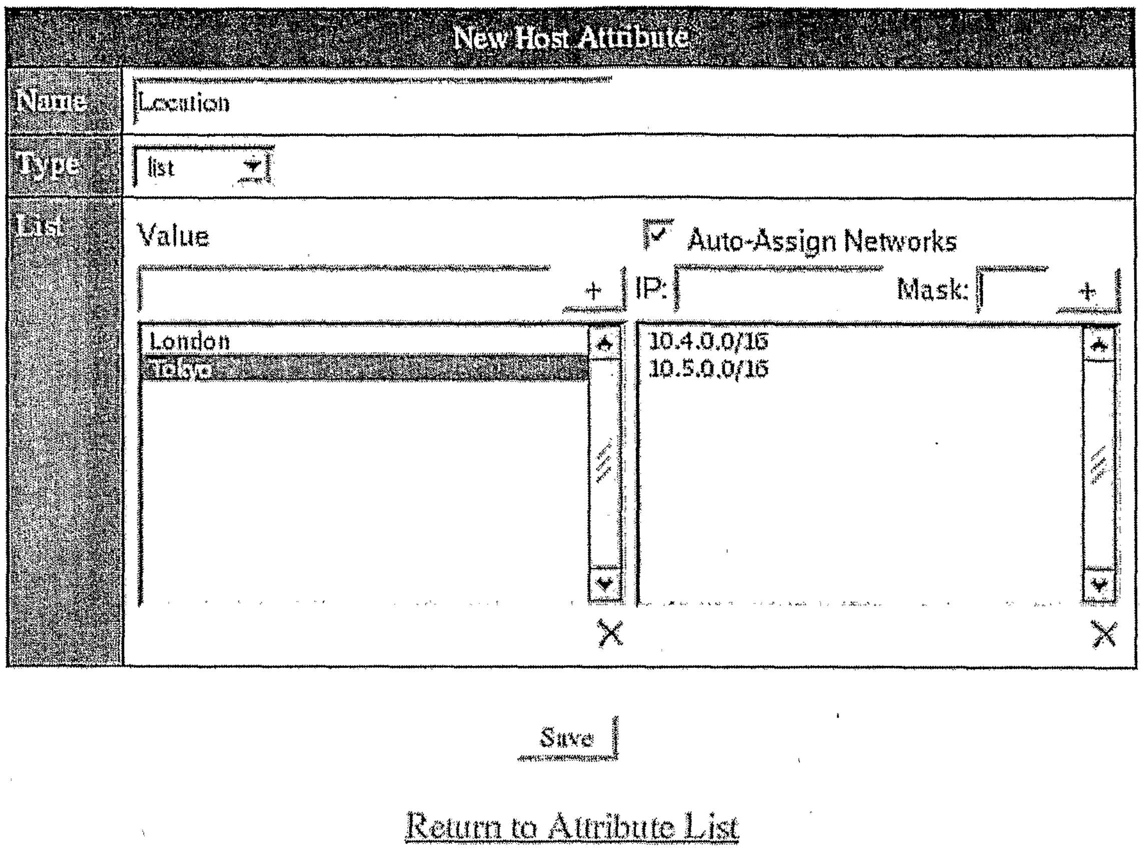
New Host Attribute	
Name	Summary
Type	text

FIG. 4



New Host Attribute	
Name	Room Number
Type	Integer
Min	1
Max	101

FIG. 5



New Host Attribute

Name: Location

Type: list

List

Value

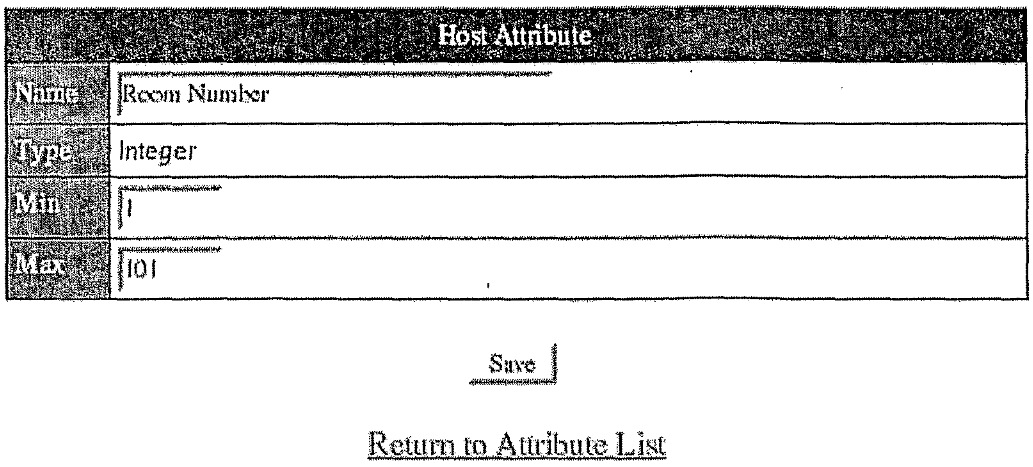
London
Tokyo

Auto-Assign Networks

IP: Mask:

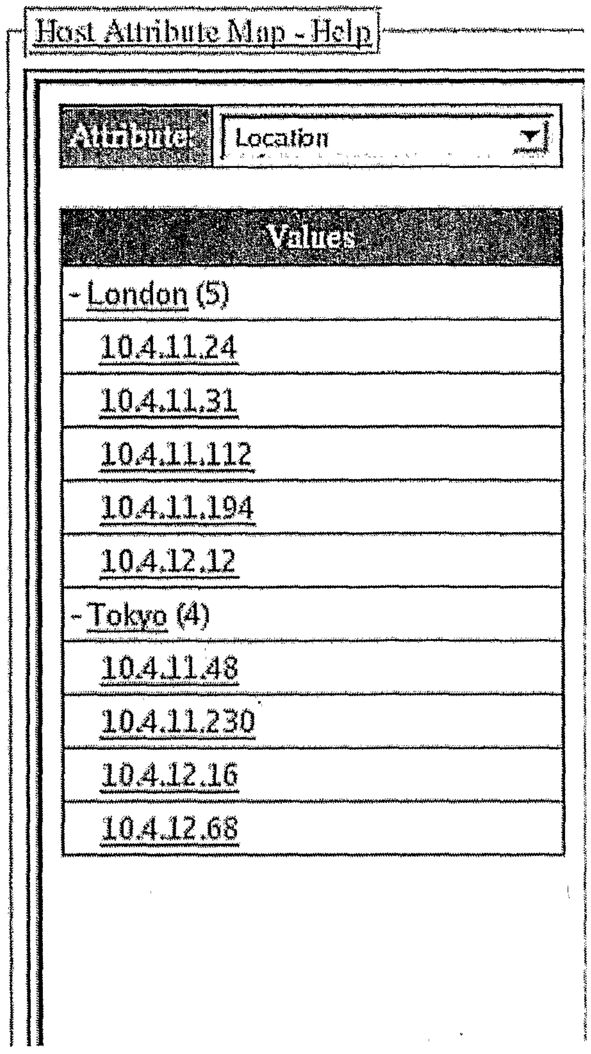
10.4.0.0/16
10.5.0.0/16

FIG. 6



Host Attribute	
Name	Room Number
Type	Integer
Min	1
Max	101

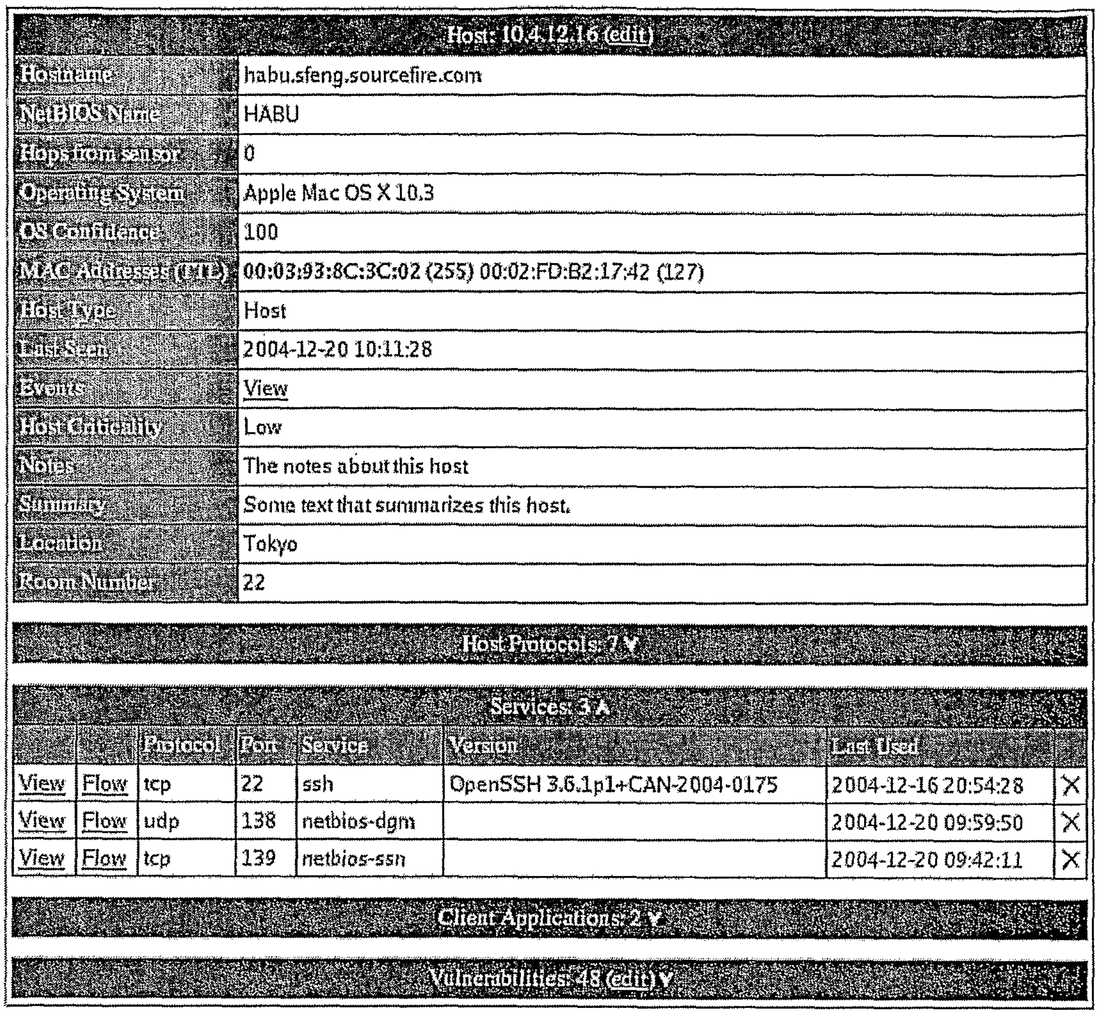
FIG. 7


Host Attribute Map - Help

Attribute: Location

Values
- London (5)
10.4.11.24
10.4.11.31
10.4.11.112
10.4.11.194
10.4.12.12
- Tokyo (4)
10.4.11.48
10.4.11.230
10.4.12.16
10.4.12.68

FIG. 8



Host: 10.4.12.16 (edit)	
Hostname	habu.sfeng.sourcefire.com
NetBIOS Name	HABU
Hops from sensor	0
Operating System	Apple Mac OS X 10.3
OS Confidence	100
MAC Addresses (TTL)	00:03:93:8C:3C:02 (255) 00:02:FD:B2:17:42 (127)
Host Type	Host
Last Seen	2004-12-20 10:11:28
Events	View
Host Criticality	Low
Notes	The notes about this host
Summary	Some text that summarizes this host.
Location	Tokyo
Room Number	22

Host Protocols: 7 ∨

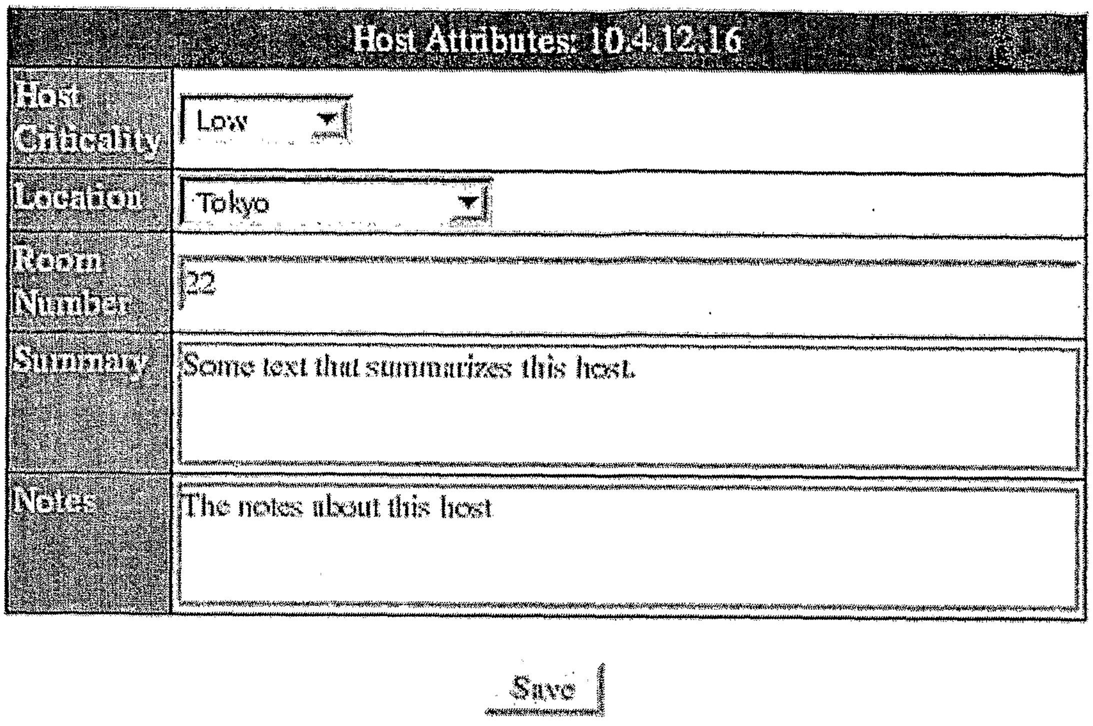
Services: 3 ∧

		Protocol	Port	Service	Version	Last Used	
View	Flow	tcp	22	ssh	OpenSSH 3.6.1p1+CAN-2004-0175	2004-12-16 20:54:28	X
View	Flow	udp	138	netbios-dgm		2004-12-20 09:59:50	X
View	Flow	tcp	139	netbios-ssn		2004-12-20 09:42:11	X

Client Applications: 2 ∨

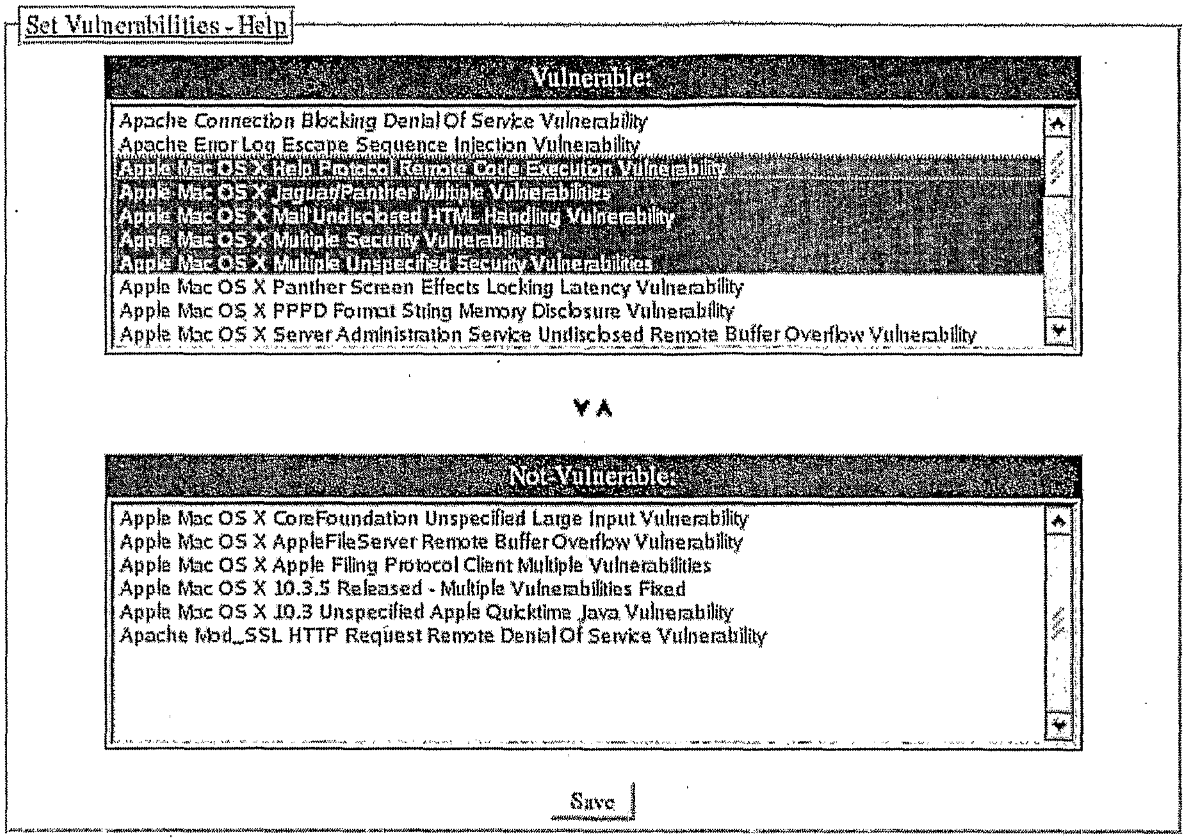
Vulnerabilities: 48 (edit) ∨

FIG. 9



Host Attributes: 10.4.12.16	
Host Criticality	Low
Location	Tokyo
Room Number	22
Summary	Some text that summarizes this host.
Notes	The notes about this host

FIG. 10


Set Vulnerabilities - Help

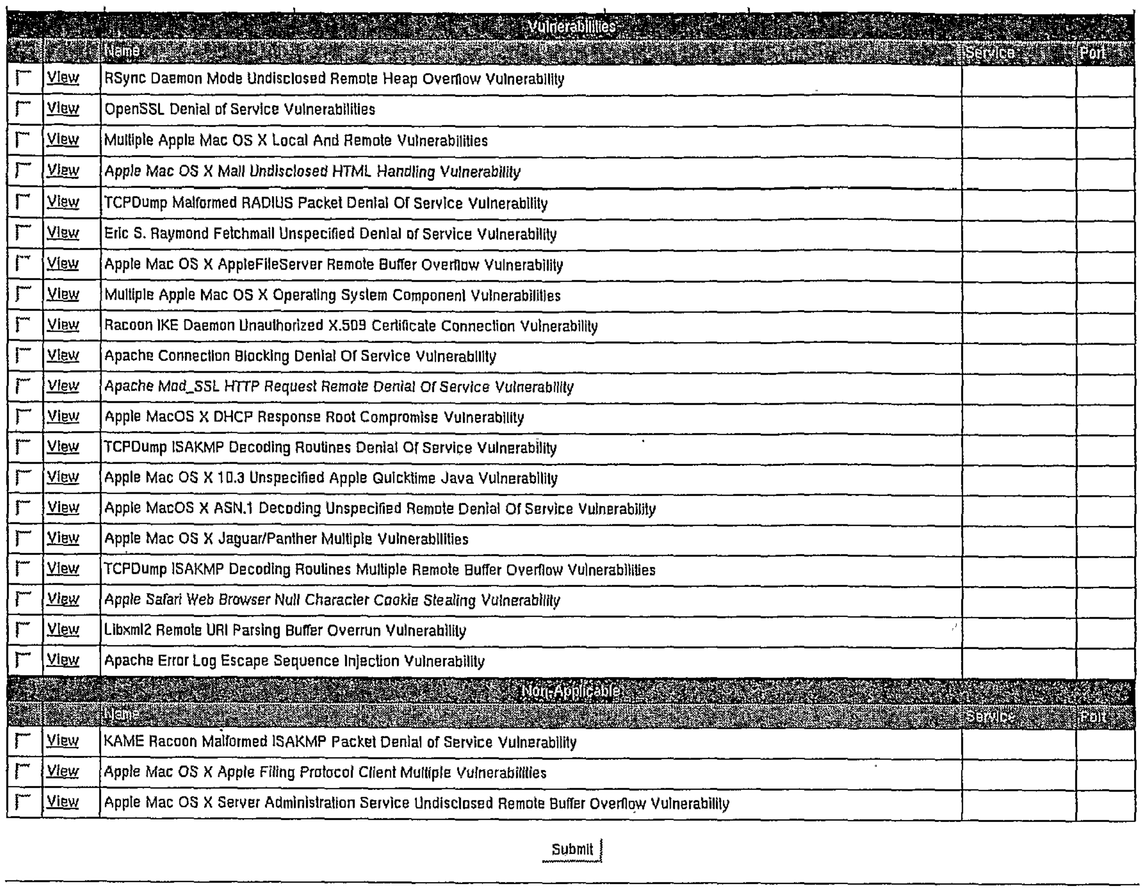
Vulnerable:

Apache Connection Blocking Denial Of Service Vulnerability
Apache Error Log Escape Sequence Injection Vulnerability
Apple Mac OS X Help Protocol Remote Code Execution Vulnerability
Apple Mac OS X Jaguar/Panther Multiple Vulnerabilities
Apple Mac OS X Mail Undisclosed HTML Handling Vulnerability
Apple Mac OS X Multiple Security Vulnerabilities
Apple Mac OS X Multiple Unspecified Security Vulnerabilities
Apple Mac OS X Panther Screen Effects Locking Latency Vulnerability
Apple Mac OS X PPPD Format String Memory Disclosure Vulnerability
Apple Mac OS X Server Administration Service Undisclosed Remote Buffer Overflow Vulnerability

Not-Vulnerable:

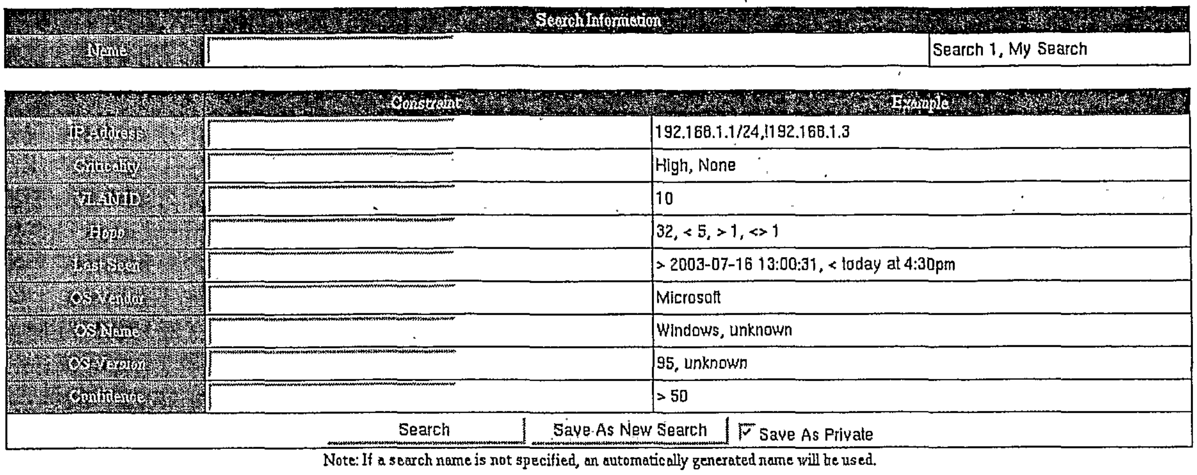
Apple Mac OS X CoreFoundation Unspecified Large Input Vulnerability
Apple Mac OS X AppleFileServer Remote Buffer Overflow Vulnerability
Apple Mac OS X Apple Filing Protocol Client Multiple Vulnerabilities
Apple Mac OS X 10.3.5 Released - Multiple Vulnerabilities Fixed
Apple Mac OS X 10.3 Unspecified Apple Quicktime Java Vulnerability
Apache Mod_SSL HTTP Request Remote Denial Of Service Vulnerability

FIG. 11



		Vulnerabilities		
		Name	Service	Port
☐	View	RSync Daemon Mode Undisclosed Remote Heap Overflow Vulnerability		
☐	View	OpenSSL Denial of Service Vulnerabilities		
☐	View	Multiple Apple Mac OS X Local And Remote Vulnerabilities		
☐	View	*Apple Mac OS X Mail Undisclosed HTML Handling Vulnerability*		
☐	View	TCPDump Malformed RADIUS Packet Denial Of Service Vulnerability		
☐	View	Eric S. Raymond Fetchmail Unspecified Denial of Service Vulnerability		
☐	View	Apple Mac OS X AppleFileServer Remote Buffer Overflow Vulnerability		
☐	View	Multiple Apple Mac OS X Operating System Component Vulnerabilities		
☐	View	Racoon IKE Daemon Unauthorized X.509 Certificate Connection Vulnerability		
☐	View	Apache Connection Blocking Denial Of Service Vulnerability		
☐	View	*Apache Mod_SSL HTTP Request Remote Denial Of Service Vulnerability*		
☐	View	Apple MacOS X DHCP Response Root Compromise Vulnerability		
☐	View	TCPDump ISAKMP Decoding Routines Denial Of Service Vulnerability		
☐	View	Apple Mac OS X 10.3 Unspecified Apple Quicktime Java Vulnerability		
☐	View	Apple MacOS X ASN.1 Decoding Unspecified Remote Denial Of Service Vulnerability		
☐	View	Apple Mac OS X Jaguar/Panther Multiple Vulnerabilities		
☐	View	TCPDump ISAKMP Decoding Routines Multiple Remote Buffer Overflow Vulnerabilities		
☐	View	*Apple Safari Web Browser Null Character Cookie Stealing Vulnerability*		
☐	View	Libxml2 Remote URI Parsing Buffer Overrun Vulnerability		
☐	View	Apache Error Log Escape Sequence Injection Vulnerability		
		Non-Applicable		
		Name	Service	Port
☐	View	KAME Racoon Malformed ISAKMP Packet Denial of Service Vulnerability		
☐	View	*Apple Mac OS X Apple Filing Protocol Client Multiple Vulnerabilities*		
☐	View	Apple Mac OS X Server Administration Service Undisclosed Remote Buffer Overflow Vulnerability		

FIG. 12



Search Information		
Name		Search 1, My Search

	Constraint	Example
IP Address		192.168.1.1/24,!192.168.1.3
Criticality		High, None
VLAN ID		10
Hops		32, < 5, > 1, <> 1
Last Seen		> 2003-07-16 13:00:31, < today at 4:30pm
OS Vendor		Microsoft
OS Name		Windows, unknown
OS Version		95, unknown
Confidence		> 50

Search | Save As New Search | [✓] Save As Private

Note: If a search name is not specified, an automatically generated name will be used.

FIG. 13

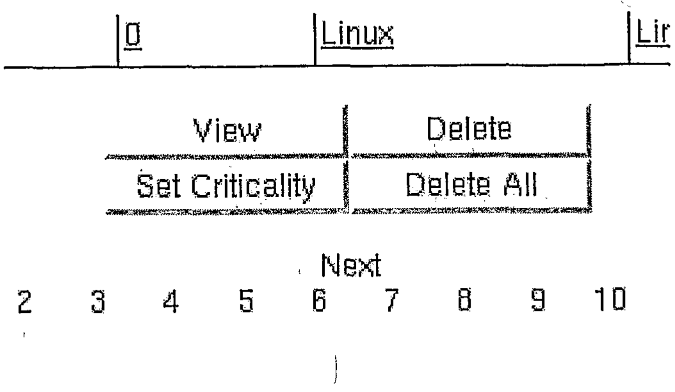
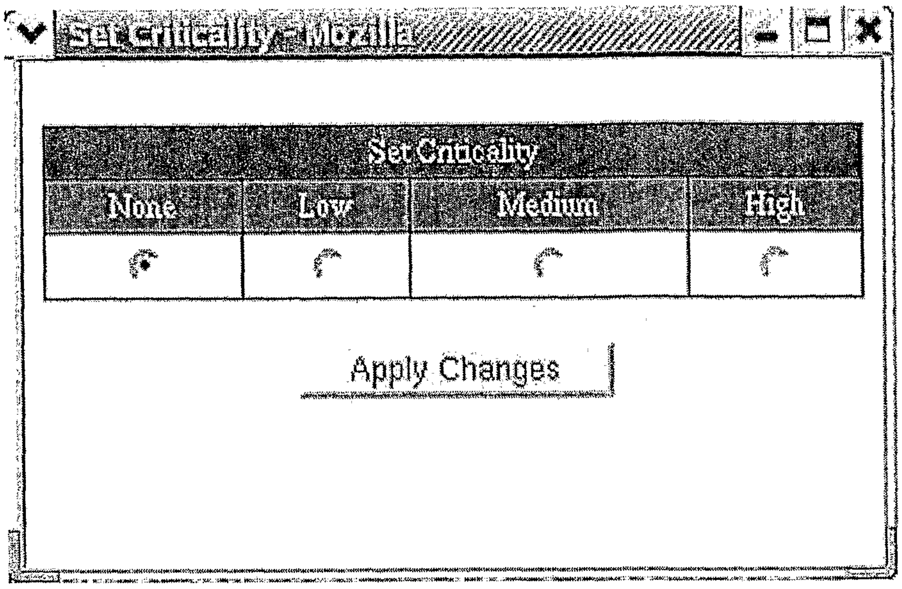
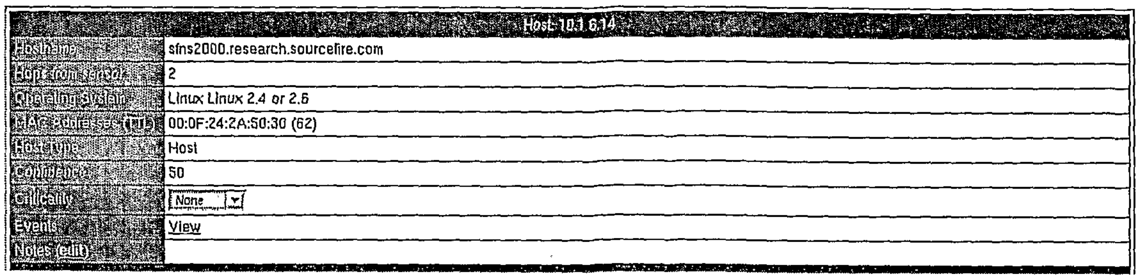

0
Linux
Lir
View
Delete
Set Criticality
Delete All
Next
2 3 4 5 6 7 8 9 10

FIG. 14


Set Criticality - Mozilla

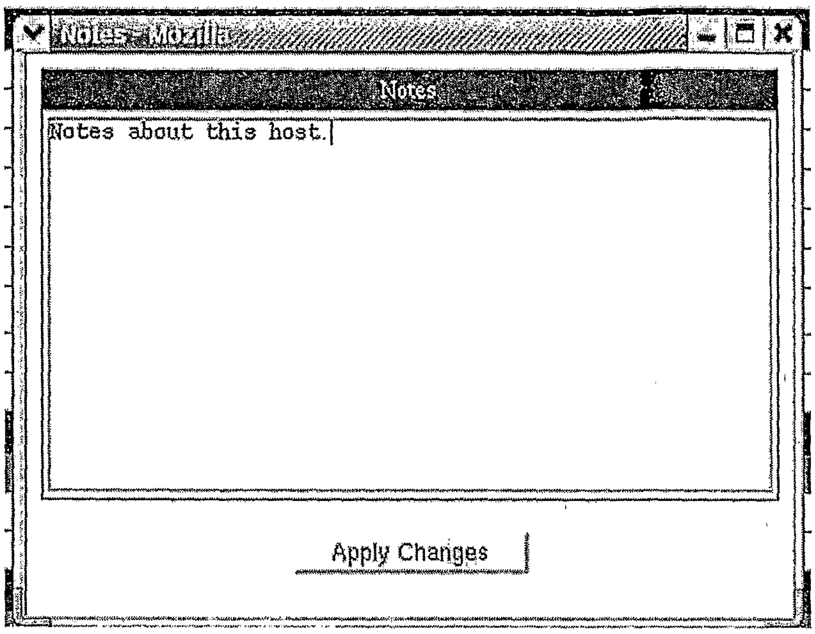
Set Criticality			
None	Low	Medium	High
●	○	○	○

FIG. 15



Host: 10.1.6.14	
Hostname	sfns2000.research.sourcefire.com
Hops from sensor	2
Operating System	Linux Linux 2.4 or 2.6
MAC Addresses (TTL)	00:0F:24:2A:50:30 (62)
Host type	Host
Confidence	50
Criticality	None
Events	View
Notes (edit)	

FIG. 16



Notes about this host.

FIG. 17