Journal of Information Processing Vol.21 No.4 632–639 (Oct. 2013)

[DOI: 10.2197/ipsjjip.21.632]

Regular Paper

Detection of Unexpected Services and Communication Paths in Networked Systems

Ichita Higurashi[1,a)] Akira Kanaoka[2,b)] Masahiko Kato[1,c)] Eiji Okamoto[3,d)]

Received: November 30, 2012, Accepted: June 14, 2013

Abstract: Gaining complete understanding of the active services and open communication paths is often difficult because of the rapidly expanding complexity of those services and their wide-ranging functions. Furthermore, the IT administrators of hand-designed systems often lack ways to identify and close unnecessary services and communication pathways. In this paper, firstly we propose an automated approach to discover all active services and the permitted communications paths in networked system. Secondly, we propose a method to detect all unexpected services and communication paths in networked system for IT system administrators. We then show how hand-designed networked systems containing such devices are prone to contain numerous unnecessary active services and communication paths, which are exploited by malicious actions such a service denial, information theft, and/or cyber espionage. The evaluation result shows the effectiveness of our proposed approach.

Keywords: topology discovery, service detection, multi-layer, network model

1. Introduction

The evolving services available on the Internet are causing numerous systems to become increasingly complex. As a result, many systems are now composed of multiple small networks that often consist of multiple servers and network devices. We call such systems "networked systems."

The management of vulnerabilities and the prevention of attack damage on a networked system are both achieved through the proper configuration of multiple components, including servers, routers, switches, firewalls, and load balancers. Even though a networked system consisting of tens of servers and devices may not be considered a large system, its complexity in terms of security can be quite high. Security is achieved by considering the interactions among each server and network device. Single-point security is insufficient for achieving safety on an entire system, and so the creation of compound points over multiple layers is required.

It is crucial to design networked systems that ensure essential communications are always available to its primary service users. To achieve system security, unnecessary communication paths and services on each server and network device in a networked system should be closed. To close unnecessary communication paths and services against the ever-increasing number of threats, a complete understanding of the functions and vulnerabilities of each server and network device is required. In reality, though, the realization of security during the design phase cannot be achieved without a thorough understanding of all functions and vulnerabilities.

Gaining a complete understanding of the active services and communications on every server and network device in present operating systems is difficult, because these servers and network devices typically include a wide variety of complex functions. This means unexpected and unnecessary communication paths probably exist in such networked systems. Furthermore, it is especially difficult to close all unnecessary services and communication paths in hand-designed networked systems. If we allow unnecessary open communication paths and services to exist in a networked system, we make it possible for sophisticated attackers to access the system for malicious purposes, such as service denial, information theft, and/or cyber espionage. For example, Poison Ivy is a backdoor program that allows attackers to access infected hosts from outside the networked system and steal important information from other devices inside the networked system.

To prevent these threats, it is necessary to detect all active services and communication paths. Typically, active services that operate without the knowledge of IT system administrators are found in software products that have not been updated to the latest version, even if those vulnerabilities are found and reported by the manufacturer. Thus, unnecessary services cause service denial, information theft, and/or cyber espionage.

Nevertheless, several tools are available for assessing the vulnerabilities of a host. Vulnerability assessment tools include Network Mapper (NMAP) [9], which can assess a target host in detail. However, such tools cannot merge information gathered from a number of assessed hosts in a coordinated fashion. Several studies are aimed at discovering networked topolo-

[1] Internet Initiative Japan Inc., Chiyoda, Tokyo 101–0051, Japan
[2] Toho University, Funabashi, Chiba 274–8510, Japan
[3] University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[a)] higurashi@iij.ad.jp
[b)] akira.kanaoka@is.sci.toho-u.ac.jp
[c)] masa@iij.ad.jp
[d)] okamoto@risk.tsukuba.ac.jp

gies [4], [5], [6], [7], but these studies have discovered only single or double layers.

To achieve system security, just discovering all active services and communication paths is not enough. We also need to detect unexpected services and communication paths. We propose a method for topology discovery in multi-layer and a method for detection of unexpected services and communication paths. By combining these methods, we can automatically detect unexpected services and communication paths on servers and network devices. We then show how many hand-designed systems are likely to possess numerous unnecessary services and communication paths that expose the systems to threats, such as service denial, information theft, and/or cyber espionage.

The topology discovery method consists of two stages. In the first stage, configuration information is gathered from all servers and network devices in a networked system. In the second stage, we connect and estimate the available communication paths between the servers and network devices in a multi-layer manner. As a proof of concept, we developed a script for gathering configuration information from servers and devices, as well as a tool for connecting to and estimating the available communication paths identified from the information gathered by our script. To discover multi-layered network topology, we use information obtained from a management information base (MIB) of objects [1] and estimate the missing information. Additionally, the networked system security quantification (NSQ) model proposed by Kanaoka et al. [8] is used to evaluate the information from multi-layered systems.

The method of detection of unexpected services and communication paths identifies all services and communication paths discovered by the topology discovery method and determines whether they are unexpected. To detect the unexpected services and communication paths, we need to prepare input data, which we compare with the results of the topology discovery.

To evaluate our proposed two-method approach, we apply our developed script and tool to a networked system with a three-tiered architecture and also a networked system with a demilitarized zone (DMZ) architecture. The results of our evaluations indicate the level of understanding of services and communication paths, as well as the execution performance of our proposed approach.

This paper is organized as follows. In Section 2, we present a network model and related topology discovery studies. Topology discovery algorithm is shown in Section 3 and we mention the method to detect unexpected services and communication paths in Section 4. Section 5 describes development of our method and experimental results. We then consider our method in Section 6. In Section 7, we discuss the limitations of our current method and network model. Finally, we conclude the paper in Section 8.

2. Related Works

2.1 Networked System Security Quantification Model

The NSQ model was proposed by Kanaoka et al. [8] as a new multifunctional networked system representation model for quantifying networked system reliability. The NSQ model contains the "layer" concept and classifies various network device functions into five layers (**Fig. 1**).

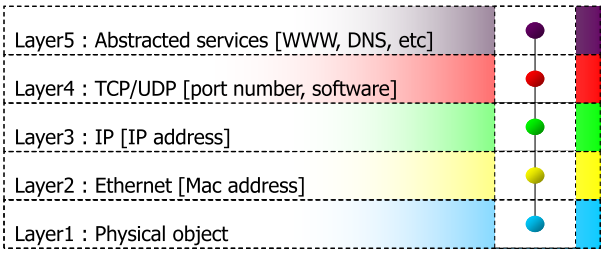

Fig. 1 Layer definition.

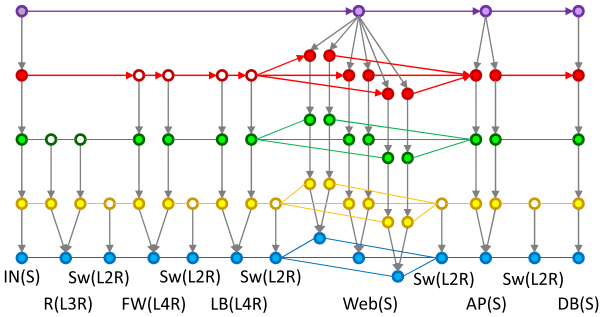

Fig. 2 Example of a networked system.

In the NSQ model, network devices and services are represented by *module*. Modules are constructed by nodes, and links corresponded to the vertex and edge in graph theory. The nodes represent communication endpoints or relay points in each layer and contain information such as a MAC address, IP address, or port number which are used as an ID. The relay nodes pass on communication data from endpoint (source) to endpoint (destination). The links represent dependency, relay, or communications between nodes.

There are seven module types: *Internet* (*I*), *Service* (*S*), *Layer 1 Relay* (*L1R*), *Layer 2 Relay* (*L2R*), *Layer 3 Relay* (*L3R*), *Layer 4 Relay* (*L4R*), and *Layer 5 Relay* (*L5R*). The Internet module represents the Internet and is the source of communication with the networked system. A service module provides services such as World Wide Web (WWW), Secure Shell (SSH), or domain name service (DNS). Relay modules represent network devices and network functions. For example, L1R is a hub, L2R is a switch, L3R is a router, L4R is a firewall and L5R is a proxy.

The NSQ model was improved to achieve more flexible and detailed expression of modules [10]. Moreover, an XML data model is also proposed using the modified NSQ model. The method of evaluating impact of vulnerabilities in a networked system is also proposed.

Figure 2 shows an example of a networked system. Use of the NSQ model allows us to explain various networked systems components as modules without losing sight of their functional characteristics.

2.2 Topology Discovery

2.2.1 Method of Breitbart et al. [4]

A method for layer 2 and layer 3 topology discovery in heterogeneous IP networks has been proposed by Breitbart et al. [4]. This method exploits the simple network management protocol (SNMP) [2], MIB objects, and the address forwarding table

(AFT). The basic idea behind this algorithm is as follows: First, the neighboring routers of a known router are discovered using routing information in MIB-II [3], and the connectivity between routers is mapped. Next, they discover the connectivity between switches, and between routers and switches, using AFTs. They then implemented their algorithm and demonstrated its ability to fully discover all paths on the target network.

2.2.2 Method of Black et al. [6]

A method for layer 2 topology discovery that does not required assistance from network devices has been proposed by Black et al. [6]. This method exploits the network behavior. More specifically, the main host instructs the other hosts in sequence to send packets and then observes whether or not the packets are delivered to each host. Utilizing this method, it is possible to obtain a layer 2 topology using an NSQ model. However, the problem is that all hosts in a network must be controlled by network managers. In order to implement this software, it is necessary for the method to install it on all hosts and to set those hosts to the promiscuous mode.

2.2.3 Method of Chen et al. [7]

A method for application dependency discovery in a networked system has been proposed by Chen et al. [7]. This method exploits the packet header and traffic delay distribution between dependent services. The basic idea behind this method is that if service A depends on service B, the delay distribution between their messages should not be random. If we use this method, we can discover topology in layer 5 of the NSQ model. However, the problem with this method is that it takes too much time to collect packets.

3. Topology Discovery Algorithm

In this section, we describe a topology discovery algorithm in a multi-layered network.

3.1 Relationship between MIB and NSQ Model

Since our approach is primarily based on SNMP, we first show how MIB objects are used to build a discovery algorithm in a multi-layered network. **Table 1** shows the relationship between the NSQ model and MIB objects. As can be seen in the table, we are unable to obtain MIB information that corresponds to portion of the nodes of layers 4 and 5, or the links of layers 1, 2, 4 and 5. Such information deficits can be resolved via MIB estimations in the next step. Details on that process will be provided in the following sections.

Even though we were unable to obtain the information of layer 1 links from the MIB object, layer 1 links are basically decided by layer 2 links. Therefore, we do not need to obtain information on layer 1 links directly.

Table 1 Relationship between the NSQ model and the MIB object.

NSQ model	MIB object
L1 Node	sysName
L2 Node	ifPhysAddress
L3 Node	ipAdEntAddr
L4 Node (listen port)	tcplocalport, udplocalport
L4 Node (transmit port)	-
L5 Node	-
L1 Link	not required
L2 Link	ipNetToNediaPhysAddress
L3 Link	ipRouteNextHop
L4 Link	-
L5 Link	-
Module Type	sysServices
Routing Information	ipForwarding

3.2 Overview

In this section, we describe a topology discovery algorithm that operates under the following assumptions:

(1) All devices support SNMP.
(2) There is no hub in a networked system.
(3) There is no Virtual Local Area Networks (VLANs) in a networked system.
(4) The AFTs are complete.

Our topology discovery algorithm is divided into two phases, *Phase 1: Information Extraction* and *Phase 2: Nodes and Links Estimation*. In phase 1, we discover the connectivity between devices using the work of Breitbart et al. [4] to obtain the configuration information of all devices. With this information, we can construct an overview of the modules and the communication links between the modules in layers 2 and 3. After information extraction, we can then estimate any missing nodes and links in layers 1, 4 and 5 in phase 2 (**Fig. 3**). As mentioned earlier, since it is impossible to obtain information on all nodes and links using MIB, estimations are made instead.

For the remainder of this paper, the word "module" indicates the data in the NSQ model that is extracted from a real "device." We refer to "devices" in phase 1, and "modules" in phase 2.

3.3 Phase 1: Information Extraction

In this phase, we discover the device information and connectivities between devices simultaneously. The basic idea behind this phase is to repeatedly find the neighboring devices of the currently known devices until no new devices can be discovered, and then to obtain configuration information from all known devices.

The initial input in our method is the IP address of a known gateway router. At the beginning, the device itself decides whether to be L3R, L4R, L5R or S using *sysServices* of MIB. If a device has sysServices (0000100) - its third bit is set, we then decide the device L3R. Depending on the device type, the relevant MIB objects are retrieved from SNMP. For example, if the device type is S, we obtain the layer 1 node that corresponds to *sysName*, the layer 2 node that corresponds to *ifPhyAddress*, the layer 3 node that corresponds to *ipAdEntAddr*, the layer 4 node that corresponds to *tcplocalport* and *udplocalport*. The services

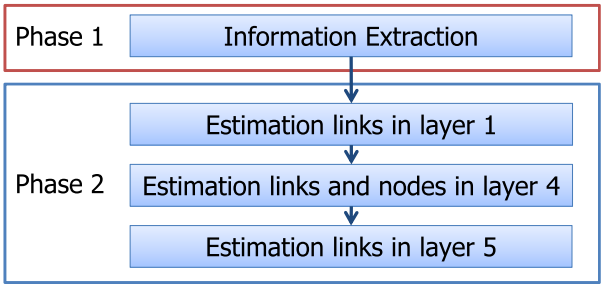

Fig. 3 Flow of the proposed method.

of the layer 5 node are determined from the port numbers of the layer 4 node. However, we cannot determine the layer 4 nodes that correspond to transmit ports at this juncture, so it is necessary to estimate them.

Second, to discover the neighboring devices from a known device, we use *ipForwarding*, which is a routing information used to discover connectivities between L3R and L3R/L4R and L4R and L4R. We then scan the subnet to discover connectivities between L3R and L5R/S, L4R and L5R/S and S and S.

After we discovered all connectivities between devices, which include L3R, L4R, L5R and S, and have obtained the configuration information of all devices, we can then discover the configuration information of the switch that corresponds to L2R, and the connectivity between switch and router as well as the switch and switch. To discover this connectivity, we apply the method of Breitbart et al.

In this way, we can recursively determine the connectivity between devices and obtain their configuration information.

3.4 Phase 2: Nodes and Links Estimation

In this phase, we estimate further nodes and links in order to complement the result from phase 1. In phase 1, we obtained device information and connectivities between the devices in layer 2 and layer 3, after which we constructed an overview of the modules. However, we were unable to obtain links for layers 1, 4 and 5 links, or the layer 4 node that corresponds to the transmit port. In phase 2, we estimate links for the links of layers 1 and 4, layer 4 nodes, and layer 5 links, in that order, using modules and layer 3 links.

First, we estimate layer 1 links. Since we have already discovered the connectivities between the devices in layer 2, we are now able to estimate the layer 1 links between modules. If layer 2 nodes a and b are connected, we can then presume that layer 1 nodes x and y, which are connected to a and b, are connected.

Next, we estimate layer 4 nodes and layer 4 links. The process used to estimate layer 4 nodes and links differs from the one used to estimate links for layers 1 because it is necessary to estimate nodes in addition to links. The basic scheme used to estimate layer 4 nodes and links is as follows: If module A and module B, which are either L4R, L5R, S or I are connected with layer 3 links and communicate with each other, those modules will have a pair of layer 4 nodes that correspond to a listen port and a layer 4 node that corresponds to a transmit port. More specifically, if module B has a layer 4 node such as 80, module A must also have a layer 4 node that corresponds to a transmit port.

In the third step, we estimate layer 5 links. If layer 4 nodes a and b are connected, we can presume that layer 5 nodes x and y, which are connected to a and b, are connected as well.

In this manner, all missing nodes and links are estimated.

4. Detection of Unexpected Services and Communication Paths

In this section, we propose a method to detect unexpected services and communication paths for IT system administrators. To the best of our knowledge, there is no way existing works to find both unexpected services and communication paths in the networked system. Hence, the method to detect them is one of our contributions. If we combine our topology discovery method with the detection method of unexpected services and communication paths, we can automatically identify the unexpected services and communication paths in a networked system.

Our proposed method detects unexpected services, communication paths, and nodes having a dependency relationship with unexpected services, and links having a dependency relationship with communication paths.

In the first step, we detect unexpected services and communication paths in a layer 5 network by comparing the input data for an ideal layer 5 network with the extracted results of the topology discovery of the real topology of the networked system. For example, if a web service is not included in the input data but the service is discovered in the extraction result, the IT system administrators can identify the web service as an unexpected service and unexpected communication paths to the unexpected web service.

In the second step, we detect unexpected nodes corresponding to ports and communication paths in layer 4. To detect unexpected nodes and communication paths in layer 4, we have to search all nodes of layer 4 and layer 5 and all communication paths in layers 4 and 5. Next, we compare the layer 4 communication paths with the layer 5 communication paths, then detect the unexpected communication paths in layer 4. If module A and module B communicate in layer 5, they also communicate in layer 4. In other words, if there is a communication path between module A and module B in layer 5, there must be a communication path between module A and module B in layer 4. Since we have detected the unexpected communication paths in layer 5 in the first step, we can detect the unexpected communication paths in layer 4 using existing communication paths in layer 5. For example, if we detect a communication path to the web service in layer 5 in the first step, we can detect a communication path to port 80 in layer 4. Comparing the communication paths in layer 4 and layer 5 in this way, we detect unexpected communication paths. Consequently, we detect unexpected nodes in layer 4. If node x of module A and node y of module B are connected by unexpected communication paths in layer 4, we identify node x corresponding to the transmit port and node y corresponding to the listen port as unexpected nodes. For example, if we identify the communication path to web services as unexpected communication paths in the second step, we also identify the transmit port and the listen port, such as port 80, as unexpected nodes. We detect unexpected nodes in layer 4 by comparing layer 4 nodes and layer 5 nodes.

In the third step, we detect the unexpected nodes corresponding to an IP address and communication paths in layer 3. The basic strategy to detect unexpected nodes and communication paths is the same as in the second step. If an unexpected communication path exists between module A and module B in layer 4, the IT system administrators can identify the communication path between them in layer 3 as an unexpected communication path. Then, if a layer 3 node has no communication path, the IT system administrators know the node is an unexpected node.

Unexpected nodes and communication paths in layer 2 and

layer 1 are detected in the same way.

In this manner, IT system administrators are able to detect all unexpected services and communication paths in the networked system.

5. Implementation and Experimental Results

In this section, we describe the research results of the Simple Network Management Protocol (SNMP), the implementation of the two methods, the experimental environment, and the experimental results.

5.1 Research of SNMP

Before attempting to execute our method, we investigated whether the network devices support the MIB objects necessary for implementation. In addition, we surveyed the SNMP software to determine whether each OS was properly configured to utilize the MIB objects. **Tables 2** and **3** provide a breakdown of the compatibility of our method with various OSs and network devices.

5.2 Topology Discovery Algorithm

5.2.1 Implementation

We implemented our information extraction and estimation methods in a Java 1.6. environment. When implementing the information extraction method, we cannot use the topology discovery method for discovering switches and the connectivities between switches and the switch and router because that method requires *complete AFTs*. We estimate the switches and connectivities between the switch and router, then implement the method to estimate layer 2 links and layer 2 routing (L2R).

5.2.2 Experimental Environment

Prior to testing our proposed method, we constructed two types of networked systems: three-tiered architecture and DMZ architecture. **Figures 4** and **5** show the two types of networked systems. These networked systems were constructed in a virtual environment by using two hosts and virtualization software.

The three-tiered and the DMZ architecture provide three services: web service, application service (AP) and database (DB) service. The web service is redundant on three servers. **Table 4** shows the specification of servers and network devices in our virtual environment. Secure Shell (SSH) service was employed to control the devices. In addition, we also installed the software to implement our method. However, we did not run any other services, nor did we change the configurations of these networked systems.

We now show the server specifications used to estimate nodes and links (**Table 5**).

The configuration of each server is initially based on installed OS related services and applications and specific services (e.g., Apache HTTPD, Apache Tomcat, MySQL). The initially installed OS services and applications sometimes run unnecessary services required for the original purpose of the server. The aim of the configuration is to expose initially installed services that cause difficulty in completely understanding a networked system and can pose a big threat to sophisticated attacks.

5.2.3 Evaluation of the Proposed Method

Table 6 shows the time required to extract information and to estimate nodes and links. We observe that information extraction took a significant amount of time; however, we also determine the reason for the long time. When we obtain MIB objects, we use the *snmpwalk* command, which enables us to retrieve all MIB ob-

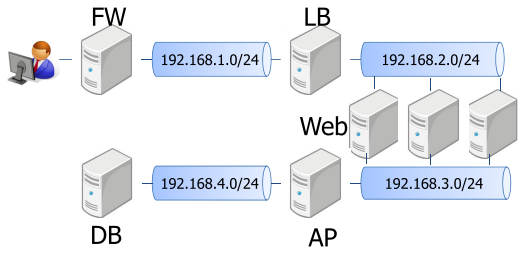

Fig. 4 Three-tiered architecture.

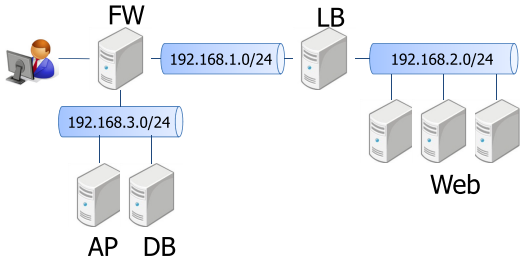

Fig. 5 DMZ architecture.

Table 2 Implementation of a MIB object.

Device Name	NSQ Model	MIB
Cisco Systems Catalyst 3560G	L2R	Available
Cisco Systems CISCO2811	L3R	Available
YAMAHA RTX1100	L3R	Available
Allied Telesis CentreCOM AR570S	L3R	Available
Juniper Networks NetScreen-5GT	L4R	Available

Table 3 SNMP software and OS compatibility.

SNMP Software	OS	MIB
NET-SNMP	Fedora 14, Ubuntu 10, CentOS 5, FreeBSD 8, Solaris 10, vyatta, Windows 7/Vista/Server 2008	Available
SNMP service	Windows 7/Vista/Server 2008	Available

Table 4 Phase 1: Experimental environment.

Device Type	LB, Web, AP, DB	Router, FW
OS	CentOS 5.8	Vyatta VC 6.4
CPU	Intel(R) Xeon(TM) CPU 5160 @3.00 GHz	
	Intel(R) Core(TM)2 Duo CPU E7400 @2.80 GHz	
Memory	512 MB	256 MB
Software	JRE 1.6	JRE 1.6
	NET-SNMP 5.3.2.2	NET-SNMP 5.6.1.1

Table 5 Phase 2: Experimental environment.

OS	CentOS 5.8
CPU	Intel(R) Core(TM) i7-3960X CPU @ 3.30 GHz
Memory	1 GB
Software	JRE 1.6

Table 6 Processing time (msec).

	Three-tiered	DMZ
Information extraction	28,693	37,408
Estimation in layer 2	172	130
Estimation in layer 1	429	316
Estimation in layer 4	5,815	2,986
Estimation in layer 5	53,827	65,921
Total	88,936	106,761

jects. Even though *snmpwalk* needs a longer time to execute than *snmpget*, *snmpwalk* can gather whole MIB objects, including network management tools with throughput data, which will be used in future applications. If we just focus on retrieving MIB objects required to build the NSQ data model, we can use the *snmpget* command and reduce the time required for information extraction since we do not need to get extra MIB objects. The method of Chen et al. [7], which is one of the method to discover the dependency of application, shows that their method takes several weeks or months because they have to collect a lot of packets. Table 6 shows that our method has the advantage of taking little time to discover network topology by comparing our method with the method of Chen.

Table 7 shows the number of nodes and links discovered during information extraction along with the estimated nodes and links. We observe that a significant number of nodes and links were estimated. Almost all node increments are layer 4 nodes that corresponded to transmit ports. Similarly, almost all link increments belong to layers 4 and 5, or the links between the layer 4 and layer 5 networks. These node and link increments indicate that the modules have a significant number of active services, and that each module can communicate with each of the other modules.

In this experiment, we constructed two types of networked systems: three-tiered architecture [11] and DMZ architecture. Both architectures are typical architectures in e-Business. Our method can use three-tiered architecture which is a more advanced version; therefore, it can discover both one-tiered architecture and two-tiered architecture.

Finally, we show a visualization of the experimental result of the three-tiered architecture in phase 1 (**Fig. 6**) and phase 2 (**Fig. 7**). Note that in this representation, we only visualize two nodes in each module, specifically FW, LB, web, AP or DB in layer 4 and layer 5, even though the method retrieves the information of 13 nodes for layer 4 from each module.

Table 7 The number of nodes and links in Phases 1, 2.

	phase 1		phase 2	
	Node	Link	Node	Link
Three-tired	461	454	1,021	31,161
DMZ	315	308	793	11,590

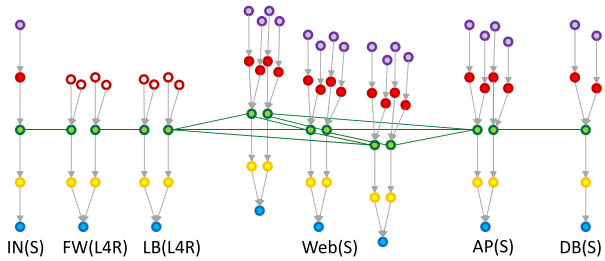

Fig. 6 Phase 1 visualization results: three-tiered architecture.

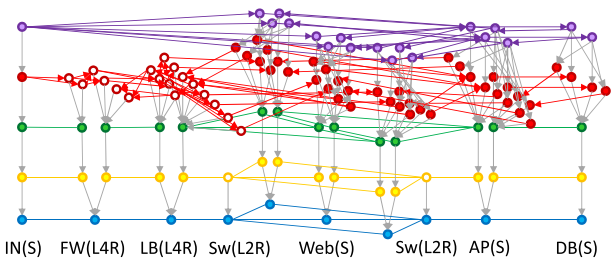

Fig. 7 Phase 2 visualization results: three-tiered architecture.

5.3 Detection of Unexpected Services and Communication Paths

5.3.1 Implementation

We implemented our method to detect unexpected services and communication paths in the Java 1.6. environment.

5.3.2 Experimental Environment

To utilize our detection algorithm, we first have to prepare an ideal layer 5 network as input data and a target network. The input data can be composed in many ways. For example, one way is to select the intended services and communication paths from a graphical user interface, and another way is to get the intended services and communication paths from the networked system specification. In this experiment, we prepare a layer 5 network represented by the NSQ model. The network consists of Apache HTTPD, Apache Tomcat and MySQL as an ideal layer 5 network (**Fig. 8**). Additionally, as a target network, we use the results of the topology discovery algorithm, that is, the three-tiered architecture (Fig. 7) and the DMZ architecture.

In the experiment of topology discovery, we recognize SSH service, snmpd service, web service, Tomcat and MySQL service as intended services. However, we recognize SSH service and snmpd service as unnecessary services in this experiment.

Table 8 show the server specifications used to detect unexpected services and communication paths.

5.3.3 Evaluation of the Proposed Method

Table 9 shows the time required to detect unexpected services and communication paths. We observe that detections of the unexpected services and the communication paths take little time.

Table 10 shows the number of nodes and links detected as the unexpected services and the communication paths. For example, we detected 951 nodes and 31,038 links in three-tiered architec-

Fig. 8 Input data: ideal layer 5 network.

Table 8 Detection environment.

OS	CentOS 5.8
CPU	Intel(R) Core(TM) i7-3960X CPU @ 3.30 GHz
Memory	1 GB
Software	JRE 1.6

Table 9 Processing time (msec).

	Three-tiered	DMZ
Detection in layer 5	28,044	8,134
Detection in layer 4	4,666	4,077
Detection in layer 3	13	82
Detection in layer 2	26	33
Detection in layer 1	13	18
Total	32,762	12,344

Table 10 Number of nodes and links in the detection algorithm.

	Intended services and communication paths		Unexpected services and communication paths	
	Node	Link	Node	Link
Three-tired	70	123	951	31,038
DMZ	111	260	682	11,330

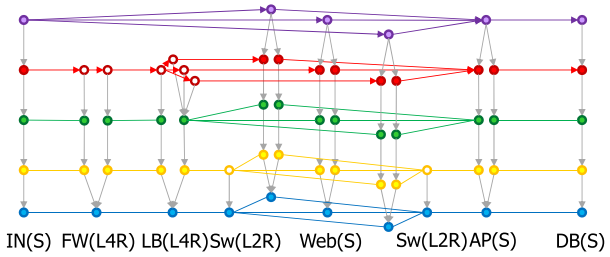

Fig. 9 Result of removing unexpected nodes and links.

Table 11 Results of web server service detection.

Port	Pro	Service	Port	Pro	Service
22	TCP	sshd	111	UDP	portmap
80	TCP	httpd	161	UDP	snmpd
111	TCP	portmap	631	UDP	cupsd
932	TCP	rpc.stad	926	UDP	rpc.statd
32,123	TCP	our software	929	UDP	rpc.statd
57,911	UDP	avahi-daemon	5,353	UDP	avahi-daemon

ture. The table also shows that three-tiered architecture generates more communication paths than DMZ architecture. Since three-tiered architecture has more communication paths in layer 3 between servers, servers can communicate with each other in layer 4 and 5, that is, nodes are fully connected.

Figure 9 shows the visualization result of removing unexpected nodes and communication paths. Nodes and links in Fig. 9 are intended services and communication paths for IT system administrators. By comparing Figs. 7 and 9, we can see there are many unexpected ones.

6. Discussion

In this section, we analyze the results of our evaluations. We conclude that the topology discovery method and the method of detecting unexpected services and communication paths offer the following advantages.

6.1 Topology Discovery Algorithm

Using the topology discovery method, IT system administrators can detect all active services and communication paths. Recently, various forms of malicious software, including Trojan horses, backdoors, computer viruses, worms, and other malware, have been used to steal important information like passwords. For example, the backdoor program Poison Ivy allows remote users (attackers) to access infected hosts. Generally speaking, if any of the devices in a networked system becomes infected with this malware, the attacker can secretly start/stop services, install/uninstall applications, remove/rename/execute files and perform other operations on that device. Then, using the infected host as a springboard, the attacker can branch out and steal important information from other devices in the system.

It is necessary to detect all active services and communication paths in a networked system to prevent information leakage. **Table 11** shows the web server services only after executing the topology discovery method. Therefore, our method enables us to detect all active services and communication paths.

6.2 Detection Algorithm

To further prevent information leakage, IT system administrators have to detect and stop unexpected services and communication paths. As a typical problem, active services that operate without the knowledge of IT system administrators are found in software products that have not been updated to the latest version, even if those vulnerabilities were found and reported by the manufacturer. Such exposed services degrade the security of networked systems.

Using our method, IT system administrators can detect the unexpected services and the communication paths, so that they can stop them. In addition, our method can detect them automatically; therefore, our method is a suitable algorithm for large networked systems consisting of hundreds of servers and network devices. Furthermore, the method can also discover full connected network topology in networked system. It is important for IT system administrators to discover the network topology, since network topologies affect the security level of networked systems. Full connected network topology such as web servers in three-tiered architecture is the worst case of the network topology, and the topology generates a lot of communication paths. In particular, if a server in that topology is infected by malware, the number of secondary infected servers are increasing by such full connectivity. Thus, we conclude that our method has a lot of effectiveness for detection of unexpected services and communication paths.

7. Limitations

In this section, we discuss the limitations of the NSQ model and our proposed methods.

7.1 Topology Discovery Method

The first limitation to the topology discovery method is related to the existence of virtual LANs (VLANs), which are very commonly used in actual networked systems. When we use our method in an actual environment having VLANs inside the networked system, we can discover layer 3 networks. Sometimes, layer 1 or 2 nodes and links might be misestimated. However, since the NSQ model itself can handle VLANs, modification of the method for VLANs might be possible.

The second limitation is related to the existence of transparent devices, such as a transparent firewall and a transparent proxy. When we use our method in an actual environment having a transparent firewall and transparent proxy, we cannot get information of these devices because the MIBs do not show up. However, in general, since these devices have administrative interfaces to manage, we can get device information from the MIB by using administrative interfaces.

The third limitation to the proposed method is that we did not conduct experiments for huge networked systems consisting of hundreds of network devices and servers. Although we are confident of the capability of our algorithm, we will never know until we try to conduct a large-scale experiment.

7.2 Multiple Ports in One Service

The limitation of this method that relates to the NSQ model is that the model cannot describe the relationships between layer 4 nodes of the same service. For example, *Tomcat* usually opens multiple ports to communicate with the web server and other

servers. If the Tomcat server is compromised through one of the ports opened by Tomcat, other ports might be used for further malicious activities. That means some layer 4 nodes are often related to one service. However, the current NSQ model does not have such an expression scheme. A relationship expression is required to use the results from the proposed method, such as the results of a vulnerability or a threat analysis of a system.

8. Conclusion

A networked system consists of servers and network devices, and it is important for IT system administrators to understand all unexpected services and communication paths operating in their networked systems. Malware is often employed to steal important information from networked systems. Active services and communication paths that are operating on networked systems without the knowledge of the administrator can cause such threats to be carried out. For this reason, discovering unexpected services and communication paths in a networked system is necessary for every IT system administrator.

In this paper, we proposed a combined topology discovery method and detection method of unexpected services and communication paths. The topology discovery method can discover all services and communication paths on every server and device in a networked system, and the detection method can identify the intended services and communication paths. By combining these methods, we can detect unexpected services and communication paths in a whole networked system. The experimental results of our method show how numerous unnecessary services and communication paths are typically left opened in a hand-designed networked system. As a result, system vulnerabilities to threats, such as service denial, information theft, and/or cyber espionage, increase. The results of the evaluation of our detection approach demonstrate its effectiveness.

References

[1] Rose, M. and McCloghrie, K.: *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC1155 (1990).

[2] Case, J., Fedor, M., Schoffstall, M. and Davin, J.: *A Simple Network Management Protocol* (*SNMP*), RFC1157 (1990).

[3] McCloghrie, K. and Rose, M.: *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, RFC1213 (1991).

[4] Breitbart, Y., Garofalakis, M., Jai, B., Martin, C., Rastogi, R. and Silberschatz, A.: Topology discovery in heterogeneous IP networks: The NetInventory system, *IEEE/ACM Trans. Networking*, Vol.12, No.3, pp.401–414 (2004).

[5] Breitbart, Y. and Gobjuka, H.: Characterization of layer-2 Unique Topologies, *Inf. Process. Lett.*, Vol.105, No.2, pp.52–57 (2008).

[6] Black, R., Donnelly, A. and Fournet, C.: Ethernet Topology Discovery without Network Assistance, *Proc. ICNP 2004*, pp.328–339 (2004).

[7] Chen, X., Zhang, M., Jai, B., Mao, Z.M. and Bahl, P.: Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions, *Proc. OSDI 2008*, pp.117–130 (2008).

[8] Kanaoka, A., Katoh, M., Toudou, N., Okamoto, E.: Extraction of Parameters from Well Managed Networked System in Access Control, *Proc. ICIMP 2009*, pp.56–61 (2009).

[9] Network Mapper, available from ⟨http://nmap.org/⟩.

[10] Harada, T., Kanaoka, A., Okamoto, E. and Kato, M.: Identifying Potentially-Impacted Area Using CVSS for Networked Systems, *Proc. CSnP 2010*, pp.367–370 (2010).

[11] Intel e-Business Center: *N-tier Architecture Improves Scalability, Availability and Ease of Integration*, White Paper (2001).

Ichita Higurashi received his B.S.Eng. and M.S.Eng. degrees in systems and information engineering from Univerisity of Tsukuba in 2011 and 2013. He has recently joined in Internet Initiative Japan Inc., Japan.

Akira Kanaoka received his Ph.D. degree in engineering from University of Tsukuba, Japan in 2004. He worked at SECOM Co., Ltd. from 2004 to 2007, and at University of Tsukuba from 2007 to 2013. He is currently an assistant professor of Department of Information Science, Faculty of Science, Toho University. His research interests include network security and cryptographic application.

Masahiko Kato recieved his B.Sc. and M.Sc. degrees in engineering from Toyohashi University of Technology in 1993 and 1995 respectively. He is now working for Internet Initiative Japan Inc. since 1998. He is currently interested in network security and cloud computing security.

Eiji Okamoto received his B.S., M.S. and Ph.D. degrees in electronics engineering from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. He worked and studied communication theory and cryptography for NEC central research laboratories since 1978. In 1991 he became a professor at Japan Advanced Institute of Science and Technology, then at Toho University. Now he is a professor at Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests are cryptography and information security. He is a coeditor-in-chief of Internatinal Journal of Information Security, and members of IEEE and ACM.