設計者の意図しないサービスと通信経路の発見手法

日暮一太 †　　金岡晃 ‡　　加藤雅彦 ł　　岡本栄司 ‡

† 筑波大学大学院 システム情報工学研究科
higurashi@cipher.risk.tsukuba.ac.jp

‡ 筑波大学大学院 システム情報系
kanaoka, okamoto@risk.tsukuba.ac.jp

ł 株式会社インターネットイニシアティブ
masa@iij.ad.jp

あらまし 複数機能を持つサーバやネットワーク機器によって複雑に構成される現代のネットワークシステムにおいて，設計者がサービスや通信経路を完全に把握し管理することは難しい．一方でこのようなネットワークシステム内での意図されていないサービスや通信経路は，DoS 攻撃や情報の窃盗などの脅威をもたらす．本論文ではこのような意図しないサービスや通信経路を発見するために，マルチレイヤでネットワークトポロジを抽出する手法を提案する．さらに提案手法の実装を行い，提案手法の有効性について示す．

Discovery of Unexpected Services and Communication Paths in Networked System

Ichita Higurashi †　　Akira Kanaoka ‡　　Masahiko Kato ł　　Eiji Okamoto ‡

†Graduate School of System and Information Engineering, University of Tsukuba
higurashi@cipher.risk.tsukuba.ac.jp

‡Faculty of Engineering, Information and Systems, University of Tsukuba
kanaoka, okamoto@risk.tsukuba.ac.jp

łInternet Initiative Japan Inc.
masa@iij.ad.jp

Abstract Gaining a complete understanding of the active services and open communication paths present in recently created networked systems consisting of various servers and network devices 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, we propose an automated approach to identifying and understanding the active services and the permitted communications on all servers and network devices. We then show how hand-designed networked systems containing such devices are prone to contain numerous unnecessary active services and communication paths, which exposes them to malicious actions such a service denial, information theft, and/or cyber espionage. An evaluation result shows the effectiveness of our proposed approach.

1 Introduction

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

Management of vulnerabilities and the prevention of attack damage on a networked system are achieved through the proper configuration of multiple aspects including servers, routers, switches, firewalls, and load balancers. 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, so the creation of compound points over multiple layers is required. Even though a networked system consisting of tens of servers and devices may not be considered a large system, its complexity in security terms can be quite high.

Designing networked systems that ensure essential communications are always available to the primary service users is crucial. Thus, unnecessary communication paths and services on each server and network device in a networked system should be closed to achieve system security. 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. Indeed, realization of security during a design phase cannot be achieved without a meticulous understanding of such functions and vulnerabilities.

However, it is difficult to gain a complete understanding of the active services and communications on every server and network device on present operating systems and network devices, because recently they have wide variety of complex functions. This means it is probable that unexpected communication paths, which are unnecessary for achieving the primary service, exist in such networked systems. Furthermore, it is especially difficult to shut down all unnecessary services and communication paths in hand-designed network systems. If we allow such 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 allow attackers to access infected hosts from outside of the networked system and steal important information from other devices in the networked system.

To prevent these threats, it is necessary to detect all active services and communication paths in a networked system. Typically, active services that operate without the knowledge of IT systems 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. Then, such unnecessarily services cause the various threats.

On the other hand, several tools are available for use in assessing the vulnerabilities of a host. Such vulnerability assessment tools include NMAP , which can assess a target host in detail. However, such tools cannot merge gathered information from a number of assessed hosts in a coordinated fashion. There have also been several studies aimed at discovering networked topologies [1, 2, 3]. However, these studies discover only single or double layers.

We propose an automated approach to understanding active services and allowed communication on servers and network devices. We then show how many hand-designed systems are likely to possess numerous unneces-

sary services and communication paths, which exposes them to threats such as service denial, information theft, and/or cyber espionage. Our proposed approach consists of two stages: The first involves gathering configuration information from each server and network device in a networked system. In the second stage, we connect and estimate available communication paths between the servers and network devices in 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 information gathered by our script. To discover multi-layered network topology, we use information obtained from management information base (MIB) objects and estimate the missing information. Additionally, the networked system security quantification model (NSQ model) proposed by Kanaoka et al [4] is used to evaluate information from multi-layered systems.

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

2 Related Works

2.1 NSQ Model

The NSQ model was proposed by Kanaoka et al. [4] 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).

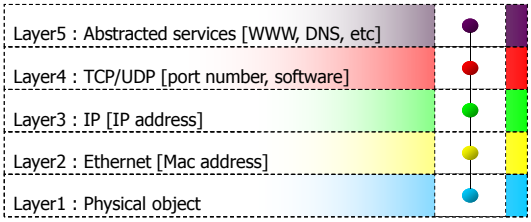

Fig. 1: Layer definition

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 IP address or port number which is used as an ID. The relay nodes pass on communication data from endpoint to endpoint. 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). Relay modules represent network devices and network functions.

2.2 Topology Discovery

A method for layer 2 and layer 3 topology discovery in heterogeneous IP networks has been proposed by Breitbart et al [1]. This method exploits the simple network management protocol (SNMP), 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, and the connectivity between routers is mapped. Next, they discover the connectivity between switches, and between routers and switches,

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

using AFTs. They then implemented their algorithm and demonstrated its ability to fully discover all paths on the target network.

3 Topology Discovery Algorithm

3.1 NSQ Model and MIB Object

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 are 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.

3.2 Overview

We describe a topology discovery algorithm that operates under the following assumptions:

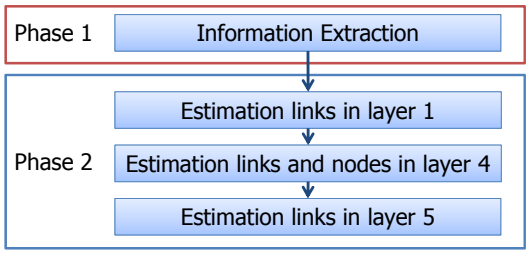

Fig. 2: Flow of the proposed method

1) all devices support SNMP; 2) there is no hub; 3) there is no Virtual Local Area Networks (VLANs); 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 [1] 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. 2). 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 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 be-

ginning, 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 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 have discovered all connectivities between devices, which include L3R, L4R, L5R and S, and have obtained the configuration information for 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 Nodes and Links Estimation

In this phase, we estimate further nodes and links in order to complement to a 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 Implementation and Results

4.1 Research of SNMP

Before attempting our method, we investigated whether the network devices support the MIB objects necessary for implementation. In

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

addition, we surveyed the SNMP software to determine whether each OS was properly configured to utilize it. Table 2, 3 provides a breakdown on the compatibility of our method with various OSs and network devices.

4.2 Implementation

We implemented our information extraction and estimation methods in the Java 1.6. environment. When implementing the information extraction method, we cannot use the previously discussed method for discovering the switches and the connectivities between switches and the switch and router because that method requires *complete AFTs*. That is why we estimate the switches and connectivities between the switch and router, and then implement the method used to estimate layer 2 links and L2R.

4.3 Experimental Environment

Prior to testing our proposed method, we constructed three types of networked systems: three-tiered architecture, DMZ architecture and, internet-router-server (IRS) architecture. Figs. 3, 4 and 5 show the three types of networked systems. These networked systems were constructed in a virtual environment using two hosts and virtualization software.

Three-tiered and DMZ architecture provide three services: Web, application service (AP) and database (DB), and Web service is got

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

redundant by 3 servers. Table 5 shows the specification of servers and network devices in our virtual environment. SSH service was employed to control the devices. In addition, we also installed the software that implemented our method. However, note that we did not run any other services, nor did we change the configurations of these networked systems.

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

Configuration of each servers is based on initially installed OS related services and applications and specific service (like Apache HTTPD, Apache Tomcat, MySQL). Initially OS installed services and applications sometimes run unnecessary services required to original purpose of the server. The aim of configuration is to expose that initially installed services cause the difficulty of complete understanding of a networked system, and might be a big threat to sophisticated attacks.

4.4 Evaluation of the Proposed Method

Table 6 shows the time required to extract information and estimate nodes and links. We observed that information extraction took a

Table 6: Processing time (msec)

	3-tiered	DMZ	IRS
Information extraction	37,912	56,402	4,193
Estimation in layer 2	381	254	102
Estimation in layer 1	572	438	185
Estimation in layer 4	33,420	23,089	341
Estimation in layer 5	49,684	196,249	210
Total	121,969	276,430	5,031

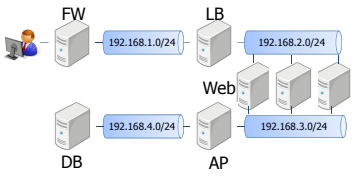

Fig. 3: Three-tiered

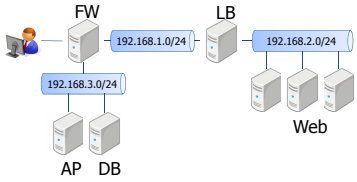

Fig. 4: DMZ

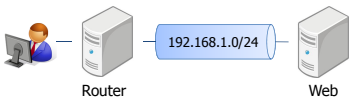

Fig. 5: IRS

Table 7: The number of nodes and links

	phase 1		phase 2	
	Node	Link	Node	Link
Three-tiered	305	312	644	5899
DMZ	204	207	453	4181
IRS	34	33	39	69

significant amount of time; however, we also managed to determine the reason why. When we obtain MIB objects, we use a *snmpwalk* command which enables us to retrieve all MIB objects.

Even though *snmpwalk* needs longer time to execute than *snmpget*, *snmpwalk* can gather whole MIB objects including objects which will be used at future application like network management tools with throughput data and so on. If we just focus on retrieving MIB objects required to build NSQ data model, we can use a *snmpget* command and reduce the time required for information extraction since we need not get extra MIB objects.

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

Finally, we will show a visualization of the experimental result of three-tiered architecture in phase 1 (Fig. 6) and 2 (Fig. 7). Note that

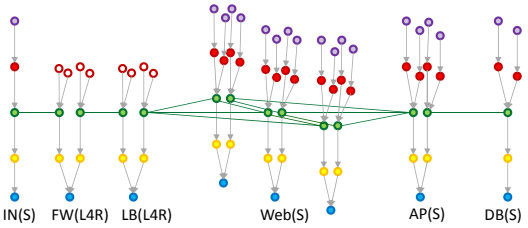

Fig. 6: Visualization results in Phase 1

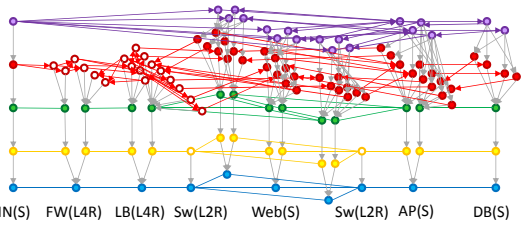

Fig. 7: Visualization results in Phase 2

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 it is possible to retrieve information on 13 nodes for layer 4 from each module.

5 Consideration

Using our method, we can detect unnecessary active services for IT system administrators. Recently, various forms of malicious software including Trojan horses, backdoors, computer viruses, worms, and other malware have been used to steal important information such as passwords. For example, poison Ivy is a backdoor program that will allow remote users (attackers) to access infected hosts. Generally speaking, if any of the devices in a networked system become infected with this malware, an attacker can secretly execute, start services, install/uninstall applications and perform other

Table 8: 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
32123	TCP	our software	929	UDP	rpc.statd
			5353	UDP	avashi-daemon
			57911	UDP	avashi-daemon

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.

To prevent information leakage, it is necessary to detect all active services in a networked system and to stop any that are unnecessary. Typically, active services that operate without the knowledge of IT systems 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.

Table 8 shows the services that our method extracted from the Web server used in the experiment. The services of *sshd*, *httpd*, *our software* and *snmpd* are those we intended to use and expected to find. However, we detected the unexpected but active services of *portmap*, *rpc.stad*, *cupsd* and *avashi-daemon*. Thus, as discussed in the sections above, our method enabled us to detect unnecessary active services.

6 Conclusion

A networked system consists of servers and network devices, and it is important for IT systems administrators to understand all the active services and communication paths that are operating in their networked systems. Recently, 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 actualization of such threats. Thus, gaining a total understanding of the active services and open communication paths in a networked system is necessary for IT systems administrators.

In this paper, we proposed an automated approach that can be used to discover and understand all the active services and open communications on every server and device in a networked system. We then showed how numerous unnecessary services and communication paths are typically left open in a hand-designed networked system, which increases system vulnerabilities to threats like service denial, information theft, and/or cyber espionage. The results of an evaluation of our automated approach demonstrated its effectiveness.

References

[1] Y. Breitbart, M. Garofalakis et al, *Topology discovery in heterogeneous IP networks: the NetInventory system*, TON, Vol. 12, No. 3, pp. 401–414, 2004.

[2] R. Black, A. Donnelly et al, *Ethernet topology discovery without network assistance*, ICNP, pp. 328–339, 2004.

[3] X. Chen, M. Zhang et al, *Automating Network Application Dependency Discovery:Experiences, Limitations, and New Solutions*, OSDI, pp. 117–130, 2008.

[4] A. Kanaoka, M. Katoh et al, *Extraction of Parameters from Well Managed Networked System in Access Control*, ICIMP, pp.56–61, 2009.

[5] T. Harada, A. Kanaoka et al, *Identifying Potentially-Impacted Area using CVSS for Networked Systems*, CSnP, pp.367–370, 2010.