An Evaluation on Feasibility of a Communication Classifying System

Yuya Sato
Graduate School of Informatics
Nagoya University
Nagoya, Japan
Email: sato.yuya@f.mbox.nagoya-u.ac.jp

Hirokazu Hasegawa
Information Strategy Office
Nagoya University
Nagoya, Japan
Email: hasegawa@icts.nagoya-u.ac.jp

Hiroki Takakura
Center for Cybersecurity
Resarch and Development
National Institute of Informatics
Tokyo, Japan
Email: takakura@nii.ac.jp

***Abstract*—Recently, sophisticated cyber attacks targeting companies or governments have frequently occurred. With conventional measures, e.g., intrusion detection system or firewalls, we cannot protect our network completely because attackers act carefully to pass through such conventional measures. Against such situation, separated network is one of the effective countermeasures. It divides an organization's internal network into multiple segments and performs fine access control among separated segments. We have proposed an automated ACL (Access Control List) generation system to support constructing separated networks previously. However, this method focuses on the business continuity of the organization, and ACL will unconditionally permit communication of a section where traffic is observed. Therefore, we proposed a communication classifying system to judge the necessity of communication and its protocol by a two-step investigation. In the first step, the system judges the consistency of the observed communication by examining the reasons why conventional systems permitted the communication. In addition, the system judges the validity of the communication by checking the waiting state of its destination terminal in the second step. In this paper, we implement the communication classifying system we have proposed, and verify the feasibility of the system. In the experiment, we applied the implemented system to a prototype network consisting of nine clients and one file sharing server (SMB (Server Message Block) protocol). As a result, our system terminated most of the unintended communication between clients and server precisely.**

Keywords–Targeted Attacks; Network Separation; Access Control.

I. Introduction

Recently, cyber attacks targeting organizations such as specific companies or countries have frequently occurred. Such attacks are called targeted attacks, and unlike indiscriminate attacks aimed at spreading simple malware, attackers attack specific organizations with sophisticated groups which have abundant funds. Therefore, attackers prepare dedicated malware for targets, and it is difficult to prevent attacks by conventional measures, e.g., firewall and intrusion detection system. Because of the above situation, recently, the focus of countermeasures has been on the mitigation of damages such as information leakage and file destruction after intrusion of malwares [1].

One of the effective countermeasures against targeted attacks is the use of a separated network [2]. It divides the organization's internal network into multiple segments and performs fine access control among the divided segments. It can prevent unintended communication among segments caused by malware, e.g., lateral movement. In addition, when we detect malwares, it can minimize the harmful effect to business continuity because we can isolate only the infected segment. However, it needs various information about networks, human resources, business contents, and so on. So, we need a large amount of cost to construct and manage a separated network.

Therefore, we have previously proposed an automated ACL (Access Control List) generation system to support constructing a separated network [3]. We call this system as "AAGS (Automated ACL Generation System)" in this paper. It generates ACL based on user's access authority to directories or files. If a user has no access authority to directories or files in a file server, the communication between the user and the file server is prohibited. However, AAGS emphasizes business continuity so that it permits all communications observed in the network even if they are unnecessary. This method may cause overly permits of unnecessary communication.

To avoid the overly permission, we have proposed a communication classifying system [4]. We call this system as "CCS (Communication Classifying System)" in this paper. CCS judges the consistency of the communication occurring in the network by analyzing the reason it was permitted. In addition, CCS judges the validity of communication which lacks consistency by using stand-by states of destination terminals. These investigations make it possible to avoid overly permitted communication.

In this paper, to verify the feasibility of CCS, we implemented AAGS and applied it to prototype network. The network is constructed with real machines, and ACL, which overly permits communication, is applied. In the experiment, our proposal correctly judged most of the overly permitted communication as unnecessary. However, there are several misjudgements by the system, and we found several problems of the system that became our future works.

In the following, Section II describes the related works. In Section III, we will explain our proposed methods, AAGS and CCS. Section IV describes the architecture of CCS, and Section V describes its implementation. The experiments using the implemented system are described in Section VI. Finally, we summarize our work in Section VII.

II. Related Works

There are many researches for preventing malware activities in internal networks. Alessandro et al. have proposed a method for modeling communication patterns of malwares that perform lateral movement [5]. However, we need large cost to employ this method because it is necessary to install a communication analysis tool on all terminals. In the case of a separated network, the spread of infection can be suppressed without installing special tools on the terminal.

Methods to construct separated network have been widely studied. Watanabe et al. proposed a VLAN (Virtual Local

 ISBN: 978-1-61208-746-7

Area Network) configuration method [6]. In this method, they monitor traffic in the network, and generate a network design by using this monitoring information. When a certain amount of traffic exceeding threshold among terminals is observed, VLAN including these terminals is generated. Because it can summarize the terminals frequently communicating with each other, it is effective from the viewpoint of amount of traffic volume. However, when a VLAN including an infected terminal is generated, it cannot prevent malware activities in that VLAN. There are many other researches to support constructing VLAN [7][8][9]. However, it is difficult to construct fine access controls among VLANs.

In addition to the above researches, there are several products, e.g., "VLAN .Config" [10], for constructing VLAN automatically. By using such products, we can construct VLAN easily, however, it is difficult to generate ACL.

III. Our Previous Research

A. Automated ACL Generation System

To support constructing networks, we proposed an automated ACL generation system (AAGS) previously [3]. AAGS judges the necessity of communication sections based on access authority of a user to files or directories in servers. If a user has no access authority to all files in the server, AAGS decides that a communication section between the user and the server is unnecessary. The system gathers information of access authorities by analyzing the information in directory service server.

In addition, AAGS analyzes the mirrored packets of the internal network. Before applying the generated ACL, the system revises it by using mirrored packets. Even if a communication was judged as unnecessary previously, its new observation calls reevaluation and then, the communication is judged as necessary. Finally, based on the judgement, the system generates ACL to permit all of the necessary communication sections. It allows us to construct a separated network easily by applying the generated ACL.

B. Problems

Because of such idea, AAGS permits all communication observed in the network even if it is an unintentionally occurring one. In other words, the system may generate ACL which overly permits unnecessary communication sections. Furthermore, the ACL generated by the system is only based on source and destination IP addresses. Once the system judged the communication section to be allowed, all communication protocols on the section are permitted.

C. Communication Classifying System

In order to solve the problems of AAGS, we proposed a communication classifying system (CCS) [4] that improves ACL generated by AAGS. CCS investigates the consistency between the communication observed in the network and the reason why AAGS permitted such communication section. If a communication lacks consistency, CCS performs additional investigation. Because appropriate ports are listened at the destination terminals if the communication is rightful, the system performs a port scan to identify the listening port and then, compares the observed communication protocols and listening ports of destination terminals. These investigations make CCS possible to detect illegal communication. In order to permit only rightful communication, the system finally generates a new ACL including prohibition of unnecessary communication sections and protocols.

D. Assumption in CCS

We proposed CCS to complement our previous AAGS. CCS assumes that the network is roughly divided into several segments, and ACL generated by AAGS is applied to the network. The applied ACL is stored in database (ACL DB) by AAGS.

ACL DB that AAGS uses is extended by adding three new columns. First, we added "Permitted Reason" to register the reason why the communication is permitted, i.e., directory service information, or communication analysis, or both of them. AAGS uses the extended versions of DB so that the ACL describes permitted communication sections, e.g., source IP addresses, destination IP addresses, and Permitted Reason. The remaining two columns are "Destination Port" and "Status". However, AAGS ignores other these two columns as empty fields.

When CCS analyzes the communication section, it registers "analyzed" to Status field of such communication section. If there is only one record for the pair of source IP address and destination IP address, and such record's Status field is empty, it is the first time for the proposed system to analyze that communication section. If "analyzed" has been registered to Status filed of a communication section, CCS omits the analysis of the communication section.

In addition, we assume that protocols are accepted by an administrator by ways different from AAGS. For example, we proposed a Dynamic Access Control System permitting communication that is overly prohibited [11]. CCS assumes that "not_analyzed" is registered to Status filed of the communication section if any other systems or administrators permit such communication. If the Status field is "not_analyzed", CCS analyzes the communication protocols in such section.

In this paper, to simplify the discussion, we assume that all terminals are statically assigned IP addresses and such assignment information is managed in a directory service server. However, our method can be easily applied to environments that employ dynamically IP address assignment method, e.g., DHCP. We can control connected device's communication by identifying the user of the device with any authentication method, e.g., IEEE 802.1X. For example, we can assign the appropriate VLAN that the user should belong to, or update ACL based on the assigned IP address.

IV. Architecture of Communication Classifying System

Figure 1 shows the architecture of CCS. The system consists of five modules and the database extended in AAGS. The details of each modules are described below.

1) Traffic Collector: This module receives all mirrored packets generated in the internal network. This paper assumes that the collection period of mirrored packets for investigation is statically defined in advance, e.g., 1 day, 1 hour, and 10 minutes. After collecting mirrored packets, the module generates a list of packet information including sets of source IP address, destination IP address, and destination port from collected packet. The generated list of packet information is sent to the Consistency Judgement module.

 ISBN: 978-1-61208-746-7

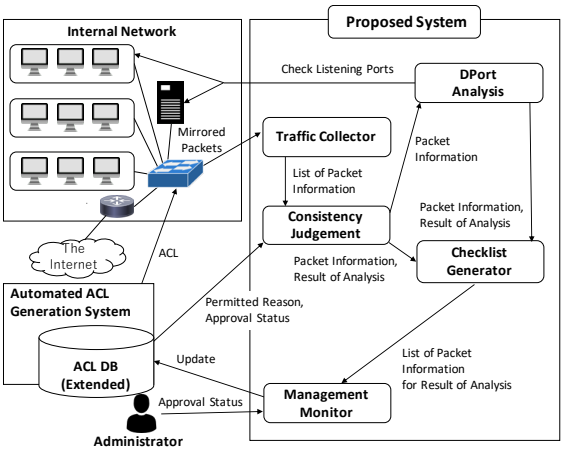

Figure 1. Architecture of Proposed System.

2) Consistency Judgment: First, when a list of packet information is received, this module searches records of ACL DB for each communication section by specifying each pair of source and destination IP addresses. When the status field is empty, the Consistency Judgment module analyzes all protocols captured in such communication section.

After extracting the subject of the communication for investigation, the Consistency Judgement module judges consistency of such communication. The module finds the permitted reason of such communication by checking ACL DB. As shown in Table I, there are six combinations of collected packet and communication reason. In the table, CA denotes communication analysis. Because AAGS checks the necessity of the file sharing communication by using a Directory Service Information (DSI), the Consistency Judgment module classifies the captured communication as SMB (Server Message Block) protocol or Other Protocols. In this paper, we assume that only SMB is used as file sharing communication protocol. SMB uses multiple ports and protocols, e.g., 139/tcp and 445/tcp. To simplify the discussion, we express these sets of all ports by using the term "SMB protocol".

TABLE I. COMBINATIONS OF PERMITTED REASON AND COLLECTED PACKET.

Collected Packet	Permitted Reason		
	DSI	DSI+CA	CA
SMB	1	2	3
Other Protocol	4	5	6

For the SMB protocol, combinations 1 and 2 of Table I have consistency. To permit these communication, the Consistency Judgement module sends this Packet Information to the Check List Generator module. On the other hand, in combination 3, communication lacks consistency, because communication of SMB protocol was observed even though there was no access authority by DSI. However, file sharing may be conducted among user's terminals directly without management by the directory service server. In order not to prohibit such communication, the Consistency Judgement module sends this Packet Information to the DPort Analysis module for additional investigation.

In case of any other protocols than SMB, only combination 4 lacks consistency. The Packet Information of such communication is sent to the Checklist Generator module to prohibit such communication. Combinations 5 and 6 have consistency, however, this module cannot determine sameness of the communication protocol collected by Traffic Collector and AAGS. The Packet Information of such communication is sent to the DPort Analysis module, which conducts a detailed investigation.

3) DPort Analysis: This module analyzes the normality of the communication. We assume that the destination terminal has to listen to the correct port of service for the communication. According to such assumption, the module judges normality of communication by using the current stand-by states of destination terminals. There are several ways to specify the listening ports of terminals, however, we adopt port-scanning against destination terminals in this paper.

Based on the result of port-scanning, when the destination port of a communication is listened on destination terminal, DPort Analysis judges that communication is necessary. On the other hand, the communication is judged as unnecessary if the destination port is blocked. Finally, these judgement results are sent to the Checklist Generator module with its packet information.

4) Checklist Generator: This module receives the packet information and judgement results from the Consistency Judgment module or DPort Analysis module. The Checklist Generator module combines these packet information and its analysis results, and generates a check list from these information for administrators. The generated check list of the packet information is sent to the Management Monitor module.

5) Management Monitor: Lists of the packet information and judgement results are sent from the Checklist Generator module to the Management Monitor module. This module presents to the administrators the combined received lists. Administrators check the list and authorize the permission or prohibition of the communication section. Finally, the module updates the ACL DB to register the authorized packet information as "analyzed" value in the status field. After updating the ACL DB, the ACL Applier in AAGS applies it to the network.

V. Implementation of Proposed System

This section describes the implementation of CSS. Figure 2 shows the basic structure of the modules and the data flow among modules. In this system, the Traffic Collector module, the Consistency Judgement module, and the DPort Analysis module run as batch processing written with Python. We adopt Node.js [12] as a Web server including the Checklist Generator module and the Management Monitor. In addition, we constructed an API server by using FastAPI [13] for smoothing data exchanges between each modules and the ACL DB.

In this paper, we implemented ACL DB and ACL Applier that are included in AAGS. We use MySQL [14] for ACL DB. By using the SDN (Software Designed Network) technique, we realized the ACL applier. We assume that Open vSwitch [15](OvS) is used as a network switch, and the SDN controller, e.g., Trema [16], instructs the OvS to control packets in the network.

In addition, all of these modules run on Docker [17], which manages applications using a container type virtual environment.

 ISBN: 978-1-61208-746-7

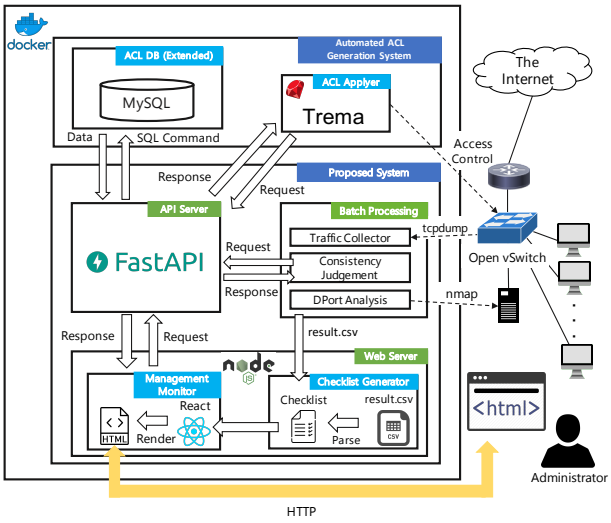

Figure 2. System Configuration Diagram.

A. Traffic Collector

This module receives mirrored packets and generates a list of packet information. We configure the OvS in advance to generate mirrors of all packets in the network and send them to the Traffic Collector. The Traffic Collector executes the `tcpdump` command and captures the mirrored packets sent from OvS for a collection period. As mentioned in Section IV, we set collection period as 10 minutes in this experiment.

The captured packets are saved as pcap files, and this module extracts sets of source IP address, destination IP address, and destination port for each packet from the pcap file by using dpkt [18], which is a module of Python. Finally, this module sends the extracted set as packet information to the Consistency Judgement module.

B. Consistency Judgement

After receiving the list of packet information, this module sends a request to the API server to search the record of communication section in the ACL DB corresponding to each packet information. In addition, this module checks the destination ports of each packet information and classifies them into SMB or other ports.

This module compares such destination ports and the result of the record search, and judges the consistency of the communication. When the module decides that the observed communication is necessary or not, it sends the packet information of that communication section with the judgement results to the Checklist Generator module. On the other hand, if the module determines that detailed analysis is necessary, it sends the packet information to the DPort Analysis module.

C. DPort Analysis

This module judges the normality of the communication that is included in packet information sent from Consistency Judgement module. To assess the listening ports of destination terminals, it uses the `nmap` command. At this time, we use the `-S` optional command of nmap to spoof the source IP address of the observed communication.

Based on the results of nmap, if the proper service port of packet information is listening at the destination terminal, the module judges this communication is rightful and it is necessary. Otherwise, the communication is judged unnecessary. After such analysis, the same way as the Consistency Judgement module determines that communication is necessary, the DPort Analysis module sends the packet information and its judgement results to the Checklist Generator module.

D. Checklist Generator and Management Monitor

The Checklist Generator module receives the packet information and its judgement results from the Consistency Judgement module and the DPort Analysis module. The Checklist Generator combines these pieces of information about the packet and generates the checklist of packet information.

The generated list of packet information is sent to the Management Monitor, and, based on this list, a html page is generated as interface for administrators by using React [19]. Figure 3 shows a sample of the generated Web page for administrators.

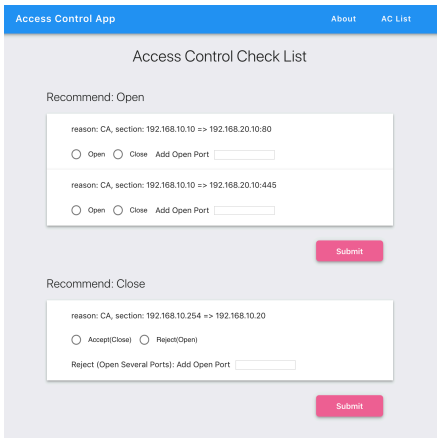

Figure 3. Sample of Management Monitor Web Page.

In this screen, there are two sections. The first section is “Recommend: Open”. The communication sections displayed in this section is judged as necessary. If the administrator judges it as appropriate, he can authorize it by selecting “Open” button. However, only the displayed ports are judged necessary by the system, and all of other ports not displayed will be prohibited. When administrators want to permit several ports in addition to the system recommendation, they can insert such ports into “Add Open Port” form. Otherwise, they use the “Close” button to prohibit the displayed communication.

The other section is “Recommend: Close”. The system judged communication displayed in this section is unnecessary. If the administrator selects “Accept (Close)” button, all communication in this section is prohibited. On the other hand, when the “Reject (Open)” is selected, the ACL permits all communication in this section. In addition, if the administrator wants to permit several ports in this section, he/she has to insert such ports into the “Add Open Port” form.

Finally, this module updates the ACL DB by using the API server after the “Submit” button is clicked. As mentioned in the next subsection V-E, the ACL DB stores only permitted communication sections. In case of that all analyzed communication is judged as still permitted, the system updated the status field of the flow_list table about such communication section as analyzed. If only several ports will be permitted,

 ISBN: 978-1-61208-746-7

in addition to the above update, those ports are inserted into dst_port field.

On the other hand, if all protocols in the communication section are judged as unnecessary, the module updates the ACL DB to delete any record of such communication section in the section_list table.

E. ACL DB (Extended)

As described in Section IV, we extended ACL DB. ACL DB consists of two tables, "section_list" and "flow_list" shown in Table II. The section_list table consists of four columns: "id", "src_ip", "dst_ip", and reason. The src_ip and the dst_ip store the source IP address and destination IP address of the communication section permitted by AAGS. The reason column stores the permitted reason.

TABLE II. ACL DB (EXTENDED) TABLE SCHEMA.

Table Name	Column	Data Type	Example
section_list	id	Integer	3
	src_ip	String	192.168.10.10
	dst_ip	String	192.168.20.20
	reason	String	CA
flow_list	section_id	Integer	3
	dst_port	Integer	443
	status	String	analyzed

The flow_list table consists of three columns that are "section_id", "dst_port", and "status". The value of the section_id is corresponding to the id of section_list table. Permitted destination ports in the communication section are stored in the dst_port column. If the communication section is permitted with no analyzation by CCS, "not_analyzed" is stored in the status column. After analyzation by CCS, the value of status is updated to "analyzed".

F. ACL Applier

We use the SDN technique to implement the ACL Applier. The OvS (Open vSwitch) is operating as core switch in the network. We use Trema as OpenFlow controller to apply the contents of ACL DB to the network.

VI. Evaluation Experiment

In order to evaluate the effectiveness of CCS, we applied the implemented system to a prototype network. We verify that if CCS can generate a ACL which prohibits the communication sections overly permitted by AAGS.

A. Experimental Conditions

1) Network Structure: For the experiment, we prepared the prototype network shown in Figure 4. The internal network is divided into three client segments according to the departments of the organization in addition to server segment.

There are three Windows 10 PCs in each segment, and all of these PCs are assigned static IP addresses, e.g., 192.168.10.10. Otherwise, in the server segment, there is only one file server assigned 192.168.100.10.

We set Open vSwitch and each segment and router are connected to this switch. In addition, Trema is assigned 192.168.200.10 and connected to the Open vSwitch directory.

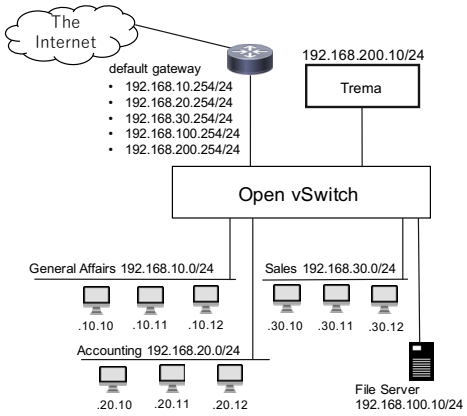

Figure 4. Proto Type Network Architecture.

2) Access Control: We assumed that AAGS generated the ACL, and we prepared the ACL shown in Table III. We configured Trema to permit only the communication listed in Table III in addition to the communications between the default gateway and all the terminals.

TABLE III. LIST OF COMMUNICATION SECTIONS PERMITTED BY PREVIOUS SYSTEM.

Source IP Address	Destination IP Address	Permitted Reason
192.168.10.10	192.168.100.10	DSI
192.168.20.10	192.168.100.10	DSI+CA
192.168.20.11	192.168.100.10	CA
192.168.30.11	192.168.100.10	CA

Although we did not prepare the directory service server in the network, the file server controls permission to files from users. In this experiment, we assume the terminals of 192.168.10.10 and 192.168.20.10 have access authority, and we insert "DSI" as Permitted Reason in the records of these communication sections.

In addition, we assume the presence of unintended communication between 192.168.20.10 and 192.168.100.10, and "CA" is added to Permitted Reason of that section. Similarly, communication sections from 192.168.20.11 and 192.168.30.11 to 192.168.100.10 are permitted because of unintended communication, and "CA" is registered as their Permitted Reason.

B. Experimental Method

The experiment was performed according to the following procedure.

Step 1: Run the proposed system and start to collect mirrored packets in the network. In this experiment, we set the collection period to be 10 minutes.

Step 2: In the collection period, terminals, i.e., 192.168.10.10 and 192.168.20.10, access the file server using the SMB protocol. In addition to these terminals, the terminal of 192.168.20.11 which has no access authority also tries SMB protocol communication with the file server. Otherwise, http protocol communication to the file server is conducted by terminals 192.168.20.10 and 192.168.30.11, although the file server does not provide http service. In addition, all nine client terminals access external sites on the Internet that are assuming activities of the organization.

 ISBN: 978-1-61208-746-7

Step 3: After 10 minutes, the collection period ends and the captured packets are analyzed by CCS. Based on the analysis result, the system generates the checklist and prepares the Web page.

Step 4: We check the result of the analysis by the proposed system on the Web page, and authorize them.

Step 5: Finally, the system applies the authorized ACL to the internal network.

C. Results of Experiment

The result of analysis by the proposed system is shown in Table IV. The legitimate SMB communication from 192.168.10.10 and 192.168.20.10 to the file server is judged as necessary correctly. In addition, the system judge the DNS protocol communication as necessary. However, it judges unintentional SMB communication between 192.168.20.11 and 192.168.100.10 as necessary.

TABLE IV. ANALYSIS RESULT BY OUR PROPOSED SYSTEM.

Internal Network Communication that Occurred			Result of
Source IP Address	Destination IP Address	Destination Port	Analysis
192.168.10.10	192.168.100.10	445	Open
192.168.20.10	192.168.100.10	445	Open
192.168.20.11	192.168.100.10	445	Open
192.168.10.12	192.168.10.254	53	Open
192.168.20.12	192.168.20.254	53	Open
192.168.30.10	192.168.30.254	53	Open
192.168.30.11	192.168.30.254	53	Open
192.168.30.12	192.168.30.254	53	Open
192.168.20.10	192.168.100.10	80	Close
192.168.30.11	192.168.100.10	80	Close
192.168.100.10	192.168.10.10	56591	Close
192.168.100.10	192.168.20.10	49977	Close
192.168.100.10	192.168.20.11	50253	Close
192.168.100.10	192.168.30.11	64131	Close
192.168.10.254	192.168.10.12	63489	Close
192.168.20.254	192.168.20.12	61236	Close
~	~	~	Close

Otherwise, the system judges several communication sections as unnecessary. It includes unintended http communication and high port number communication which seem to be returned packets.

D. Discussion

From the experimental result, we found that the proposed system correctly judged legitimate communication as necessary, i.e., SMB communication from 192.168.10.10 and 192.168.20.10 to 192.168.100.10. DNS concerned communication from clients to router is also judged as necessary correctly. In addition, unintended communication, i.e., http communication, is judged as unnecessary.

However, as the result of SMB communication between 192.168.20.11 and 192.168.100.10 shows, the proposed system misjudges the necessity of communication in the specific condition like this. This result shows the problem of CCS. In this case, the DPort Analysis module analyzed the standby state of 192.168.100.10 and judged it as necessary because 192.168.100.10 is a file server and it listened to SMB protocol ports for legitimate communication. So, the proposed system permits communication if the port of the destination terminal is opened although the communication is unintended.

Such problem is not only in the case of the SMB protocol, but it occurs in all services in which servers distinguish legitimate users by an authentication process. For example, if an unauthorized terminal attempts to access a Web server with login authentication, CCS allows this unintended communication because the HTTP and HTTPS protocols are listened in the Web server. Even when a service is provided to limited users in the same network, the proposed system makes a misjudgement and allows the communication.

If access controls are performed at terminals, this problem may not occur. To validate it, we conducted a further experiment. We set iptables at 192.168.100.10 to reject all communication not from 192.168.10.10 or 192.168.20.10, and ran CCS under that condition. In this further experiment, SMB communication between 192.168.20.11 and 192.168.100.10 is correctly judged as unnecessary.

Another solution is prohibiting access from users who have authentication process. By using authentication logs of each service, we can check whether the user has succeeded in authentication or not.

In addition to the above problem, the system displayed a lot of communication judgement between all client terminals and the router which is the default gateway of each segment. All these communications look like returned packets. We should not prohibit the returned packets, so these communications should be ignored by the system. However, if CCS simply permits all high port numbers communication, malware's communication using high port numbers is also permitted. Therefore, we need a method to distinguish whether high port communication is legitimate or not.

VII. Conclusion

In this paper, we implemented our proposed communication classifying system and applied it to a prototype network. In the experiment, the system judged necessity of most of the communication observed in the network correctly. As a result, it was confirmed that the feasibility of the proposed system, and most of the "overly permits of unnecessary communication" that was a problem in our previous proposal Automated ACL Generation System could be prohibited.

However, we found several problems from the experimental result. First, our proposed system judges unintended communication as necessary in a specific condition. When a communication occurs in the internal network and destination terminal provides service related to such communication, the CCS permits the communication unconditionally. In addition, the experimental result includes a lot of communication that is returned packets. To ignore such returned packets, we have to classify the communication as malware's activity or returned packet.

As future work, we have to propose methods to avoid misjudges and to classify the high destination port communication.

References

[1] P. Cichonski, T. Miller, T. Grance, and K. Scarfone, "Computer Security Incident Handling Guide," NIST Special Publication 800-61 Revision 2, 2012, NIST SP800-61 Rev.2.

[2] J. Information-technology Promotion Agency, "Design and Operational Guide to Protect against "Advanced Persistent Threats" Revised 2nd edition," 2011, URL: https://www.ipa.go.jp/files/000017299.pdf [accessed: 2019-09-03].

[3] H. Hasegawa, Y. Yamaguchi, H. Shimada, and H. Takakura, “An Automated ACL Generation System using Directory Service Information and Network Traffic Data (in japanese),” The IEICE Transactions on Information and Systems (Japanese Edition), vol. J100–D, no. 3, 2017, pp. 353–364.

[4] Y. Sato, H. Hasegawa, and H. Takakura, “Construction of Secure Internal Networks with Communication Classifying System,” Proceedings of the 5th International Conference on Information Systems Security and Privacy, vol. 1, 2019, pp. 552–557.

[5] G. Alessandro, P. Giovanni, C. Alberto, and B. Giuseppe, “Advanced widespread behavioral probes against lateral movements,” International Journal for Information Security Research, vol. 6, 2016, pp. 651–659.

[6] T. Watanabe, T. Kitazaki, T. Ideguchi, and Y. Murata, “A Proposal of Dinamic VLAN Configuration with Traffic Analyzation and Its Evaluation Using a Computer Simulation (in Japanese),” IPSJ Journal, vol. 46, no. 9, 2005, pp. 2196–2204.

[7] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, “Resonance: Dynamic Access Control for Enterprise Networks,” Proceedings of the 1st ACM SIGCOMM 2009 Workshop on Research on Enterprise Networking, 2009, pp. 11–18.

[8] T. Miyamoto, T. Tamura, R. Suzuki, H. Hiraoka, H. Matsuo, and et al., “VLAN Management System on Large-scale Network (in Japanese),” Transactions of Information Processing Society of Japan, IPSJ Journal, vol. 41, no. 12, 2000, pp. 3234–3244.

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, and M. Casado, “Nox: Towards an operating system for networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 3, 2008, pp. 105–110.

[10] “VLAN .Config,” 2019, URL: http://www.iiga.jp/solution/config/vlan.html [accessed: 2019-09-03].

[11] S. Nakamura, H. Hasegawa, Y. Tateiwa, H. Takakura, Y. Kim, and et al., “A Proposal of Dynamic Access Control with SDN for Practical Network Separation,” IEICE Technical Report, vol. 117, no. 299, 2017, pp. 65–69.

[12] “Node.js,” 2019, URL: https://nodejs.org/ [accessed: 2019-09-03].

[13] “FastAPI,” 2019, URL: https://fastapi.tiangolo.com [accessed: 2019-09-03].

[14] “MySQL,” 2019, URL: https://www.mysql.com [accessed: 2019-09-03].

[15] “Open vSwitch,” 2019, URL: https://www.openvswitch.org [accessed: 2019-09-03].

[16] “Trema,” 2019, URL: https://trema.github.io/trema [accessed: 2019-09-03].

[17] “Docker,” 2019, URL: https://www.docker.com [accessed: 2019-09-03].

[18] “dpkt,” 2019, URL: https://dpkt.readthedocs.io/en/latest/ [accessed: 2019-09-03].

[19] “React,” 2019, URL: https://reactjs.org [accessed: 2019-09-03].

 ISBN: 978-1-61208-746-7