Policy-based Detection and Blocking System against Abnormal Applications by Analyzing DNS Traffic

Hikaru Ichise
Tokyo Inst. Tech. Tokyo, Japan
hichise@nap.gsic.titech.ac.jp

Yong Jin
Tokyo Inst. Tech. Tokyo, Japan
yongj@gsic.titech.ac.jp

Katsuyoshi Iida
Hokkaido Univ. Sapporo, Japan
iida@iic.hokudai.ac.jp

***Abstract*—Bot-infected computers, which are compounded by botnet communication, conduct botnet-based cyber attacks using various application protocols. When using legitimate applications, a computer mostly performs domain name resolutions via the DNS full-service resolver of the organization network in advance for further communication with the application servers. During the domain name resolution, a DNS full-service resolver at least obtains the DNS NS (Name Server) records, the corresponding glue A records (IP address of the Name Server), and the application specific records, such as MX (Mail Exchange) record in case of mail transmission using Simple Mail Transfer Protocol (SMTP) of the target domain name. On the other hand, bot-infected computers with abnormal applications directly communicate with the application servers without obtaining these DNS records so that direct outbound application traffic will be generated. In this paper, we focus on this kind of direct outbound application traffic and propose a policy-based detection and blocking system against abnormal applications by analyzing DNS traffic. Specifically, the direct outbound application traffic without corresponding domain name resolutions will be detected and blocked as abnormal network traffic from bot-infected computers. We implemented a prototype system and conducted the feature evaluation on the SMTP protocol. The results confirmed that the proposed system worked correctly as designed.**

***Index Terms*—Botnet, abnormal application traffic, DNS, RPZ, SMTP, SIP, SDN, direct outbound communication.**

I. Introduction

Under the influence of COVID-19, many network applications have become popular among the Internet users for working at home nowadays. Meanwhile, the Internet users are still exposed to the risk of specific cyber attacks caused by botnets. For example, the number of malicious emails, which is one of the typical attacks of botnets, has increased drastically in 2022 [1]. However, since email transmissions also include legitimate communication, it is difficult for network administrators to simply block email applications.

Botnet has become one of the major threats to organization networks [2]. Figure 1 illustrates an overview of the botnet basis cyber attack scenario. First, a computer (end terminal or server) is infected by some kinds of malware via email attachments and browsing, etc. (arrow 1). Then the bot-infected computer searches and communicates with the Command and Control (C&C) servers for receiving instructions and conducting cyber attacks such as sending spam/phishing emails (arrow 2). This process is named

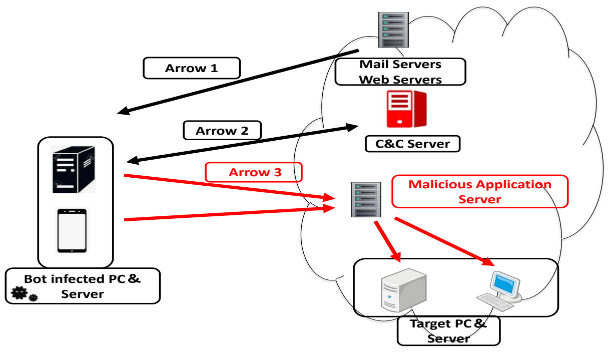

Fig. 1. Overview of botnet communication

botnet communication and it can use several communication protocols such as IRC, HTTP, P2P, and DNS [3], [4]. Finally, by following the instructions received from C&C servers, the bot-infected computer conducts cyber attacks such as spam mail distribution, malicious VoIP calls, etc, to the target victims (arrow 3). The procedure looks simple but botnet-based cyber attacks involve a huge number of bot-infected computers so that the victims consequently suffered heavy damage. So far, many approaches have been proposed to detect and block botnet communication [3], [5]–[9].

However, it is not easy to identify the botnet communication at the early stage of botnet-based cyber attacks. Thus, in this paper, we consider a detection and blocking method against abnormal applications by analyzing the DNS traffic from bot-infected computers. In general, legitimate applications perform domain name resolutions for the specific DNS records, such as MX (Mail eXchange) records in case of an email application, via a DNS full-service resolver in prior to further communication. Specifically, on receiving the DNS query from a computer (on which the legitimate applications are installed), the DNS full-service resolver obtains DNS NS (Name Server) records and the corresponding glue A records of the target domain first. Then DNS full-service resolver queries the specific record (e.g., MX record) to the name server and replies the results back to the computer. On the other hand, abnormal applications are not likely to conduct the same procedure as the above. Instead, the abnormal applications communicate with the application servers directly without the corresponding domain name resolutions. Based on this characteristic, our key idea of this research is that the analysis of DNS traffic from each internal computer will lead to identifying the abnormal applications effectively.

To implement the idea, in this paper, a way of creating the whitelist for the achieved NS as well as the corresponding glue A records is considered and a policy-based detection and blocking system against abnormal applications by analyzing DNS traffic is proposed. In the proposed system, only the applications that conduct the corresponding previous domain name resolution will be allowed for further communication, otherwise, they will be detected and blocked as abnormal applications. We implemented a prototype system by providing the whitelist functionality using DNS Response Policy Zone (RPZ) [10] which is a feature of BIND and evaluated the feature on an SDN-based local network environment. The evaluation results confirmed that the prototype system worked correctly for the application protocols, SIP and SMTP.

The rest of this paper is organized as follows. In section II, we introduce the definition of abnormal application in this paper and the related research. Then we describe the proposed method in section III and the prototype implementation in section III and IV, respectively. After that, we describe the discussion regarding some potential issues in the proposed system in section V and finally, we conclude the paper in section VI.

II. Abnormal Applications and Related Researches

As stated in the Introduction, the objective of this research is to devise a method to detect and block abnormal applications such as spam mail distributors and malicious VoIP programs. In this section, we describe the scope of abnormal applications considered in this paper and the related research.

A. Abnormal applications

Before presenting the abnormal application process, we describe a brief communication process in the legitimate application using Fig. 2. First, the client sends the DNS query to the DNS full-service resolver in the organization network for domain name resolution (arrow 1). Then the DNS full-service resolver communicates with the authoritative DNS servers for obtaining the NS record, the corresponding glue A record, and the specific application related DNS records (arrows 2 and 3). For example, the DNS full-service resolver receives MX records in the case of SMTP [11] communication, and SRV and NAPTR records in the case of SIP [12] communication as the specific application related DNS records from the authoritative DNS servers, respectively. There will be likely to extend specific application related DNS records for other applications such as HTTPS. After that, the client receives the DNS response (arrow 4) and communicates with the application server on the Internet (arrows 5 and 6).

In contrast, Fig. 3 depicts that an abnormal application has the process without conducting the corresponding previous

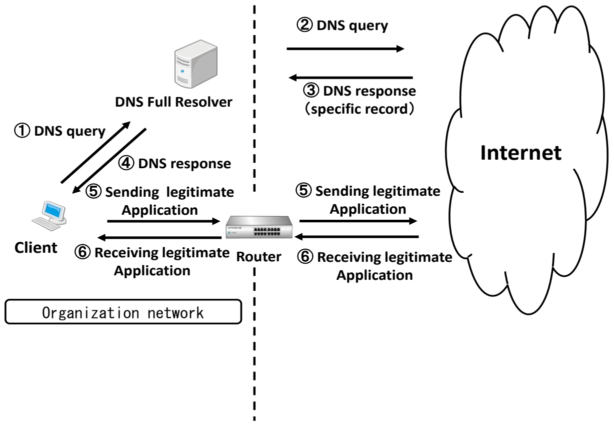
② DNS query
DNS Full Resolver
③ DNS response (specific record)
① DNS query
④ DNS response
Internet
⑤ Sending legitimate Application
⑤ Sending legitimate Application
Client
⑥ Receiving legitimate Application
Router
⑥ Receiving legitimate Application
Organization network

Fig. 2. Legitimate Application Process

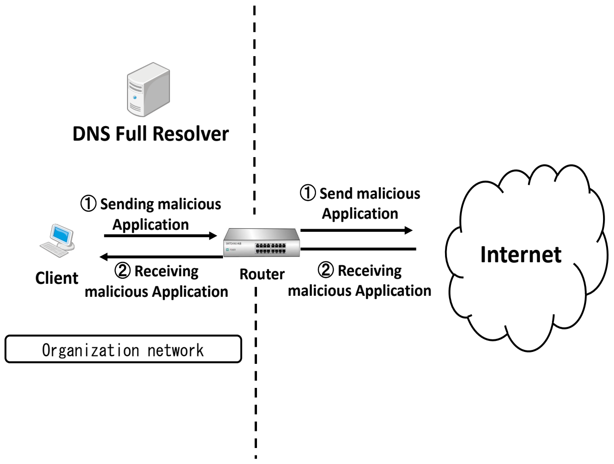
DNS Full Resolver
① Sending malicious Application
① Send malicious Application
Internet
Client
② Receiving malicious Application
Router
② Receiving malicious Application
Organization network

Fig. 3. Abnormal Application Process

DNS domain name resolution. The client infected with malicious applications directly communicates with the malicious application server, which is named a direct outbound application in this paper, for stealth behavior (arrows 1 and 2). The clients transmit spam emails and malicious VoIP messages based on the applications. Thus, it is essential to detect and block these kinds of direct outbound applications at the early stage.

It should keep in mind that the client can also receive incorrect DNS responses by the hijack of DNS full-service resolvers and authoritative DNS servers. In this paper, it is important to process the orthodox DNS name resolution by focusing on the NS records, the corresponding glue A record, and the specific application related DNS records in advance. Therefore, there are beyond the field of this paper in the cases of the hijack of DNS full-service resolvers and authoritative DNS servers and we do not include the detailed discussion here. Consequently, we consider that in the case of collecting obtained legitimate NS records, including the corresponding glue A records and the specific application related DNS records, and confirming the destination IP addresses of all the application servers on the Internet, these malicious applications will be detected and blocked. Therefore, it will be critical to constitute a system to detect and block these malicious application behavior.

B. *Related researches*

Many researches have been done in the literature for detecting and blocking botnet-based cyber attacks. In [13], the authors devised a detection system for worm infected computers sending mass-mailing by analyzing DNS traffic using the Bayesian method. The DNS traffic was obtained in about two hours from ISPs (Internet Service Providers). In particular, the A record and the corresponding MX record were analyzed using the Bayesian method. As a result, the proposed system achieved a reduction of 89% in MX records. In [14], the authors focused on malicious SMTP communication sending direct email without obtaining an MX record. The detection technology can detect mass-mailing activity from clients. Moreover, in [14], the authors focused on malicious SIP servers and aim to detect malicious VoIP messages using machine learning technology.

However, none of the existing researches considers detecting and blocking malicious applications at the early stage by analyzing DNS traffic before the communication of real attacks happens. In this paper, we intend to establish the detection and blocking system of various malicious applications for the consideration of network administrators effectively.

III. Proposed System

In this section, we give an account of the scheme of the proposed system using RPZ first. Then, we explain the system architecture to detect and block malicious traffic generated by bot-infected computers using direct outbound applications non-via DNS full-service resolver in an organization network.

A. *Overview of proposed system based on DNS RPZ*

DNS RPZ is a special authoritative zone in a DNS server for policy-based control of the target domain name and it is a special feature of BIND. Figure 4 shows a brief procedure of domain name resolution involving DNS RPZ. In a normal name resolution process without involving DNS RPZ, the client sends a DNS query to the DNS full-service resolver first. Then the DNS full-service resolver queries to the authoritative DNS servers iteratively and replies the answer to the client. Finally, the client receives the DNS response from the DNS full-service resolver.

On the other hand, in the name resolution process involving DNS RPZ, the client sends the DNS query to the DNS full-service resolver first. Then the DNS full-service resolver replies directly to the answer from its RPZ instead of querying the authoritative DNS servers iteratively. By using DNS RPZ in the proposed system in reality, a direct outbound DNS query can be regulated based on the policies effectually. The proposed system is founded on the following three observations.

(1) Any destination IP addresses of orthodox applications should be obtained in accordance with orthodox DNS based domain name resolution including the NS records, the corresponding glue A records, and the specific application related DNS records.

(2) The abnormal application defines as the direct outbound application traffic in which destination IP addresses are not obtained in accordance with legitimate DNS based domain name resolution.

(3) Some exceptions have the utility of public DNS servers.

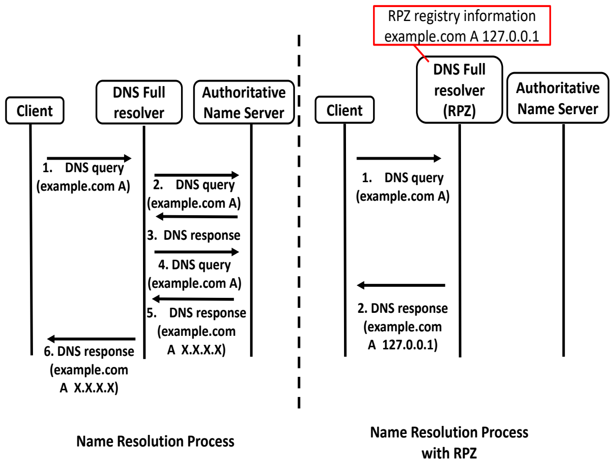
RPZ registry information
example.com A 127.0.0.1
Client
DNS Full resolver
Authoritative Name Server
1. DNS query (example.com A)
2. DNS query (example.com A)
3. DNS response
4. DNS query (example.com A)
5. DNS response (example.com A X.X.X.X)
6. DNS response (example.com A X.X.X.X)
Client
DNS Full resolver (RPZ)
Authoritative Name Server
1. DNS query (example.com A)
2. DNS response (example.com A 127.0.0.1)
Name Resolution Process
Name Resolution Process with RPZ

Fig. 4. Overview of DNS RPZ

Figure 5 illustrates the overview of system architecture in an organization network including the proposed system. In general, most of the internal computers in an organization network use the internal DNS full-service resolver for domain name resolutions while some of them use public DNS servers which cause direct outbound DNS queries. However, this kind of direct outbound DNS traffic will be treated as legitimate since the purpose is domain name resolution.

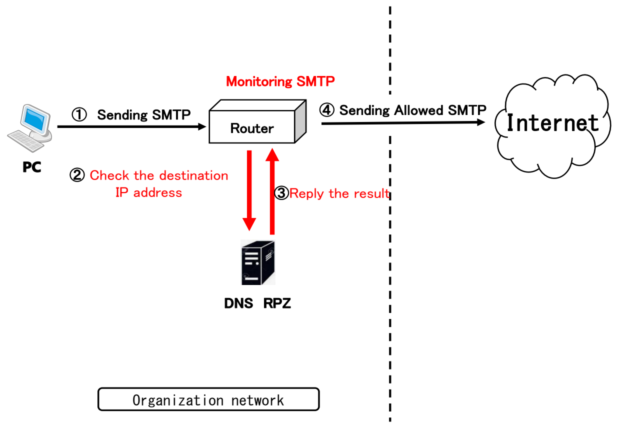

Fig. 5. Overview of proposed system

The proposed system captures and analyzes all DNS traffic generated in the organization network, and saves the NS records, the corresponding glue A records, and the MX records (in the case of email applications) to the DNS RPZ. Then all the network traffic will be monitored in the organization network and the destination IP addresses will be checked in the DNS RPZ. If the destination IP addresses are stored in the DNS RPZ, the SMTP traffic will be passed through, otherwise, the SMTP traffic will be blocked.

B. Data plane and Controller in SDN

Software-Defined Networking (SDN) states one of the new network technologies to programming language by providing the feature to brief network behavior management by open interfaces flexibly. SDN consists of the data plane and the control plane. The control plane is forwarded the information from the data plane. In other words, SDN enables network administrators to develop a program in order to modify the path of all network traffic easily. With these capabilities, we can verify if the packets transmitted from the client are a DNS query and the way to manipulate them. This task can realize by SDN technology as to isolating the routine of a switch or router into 3 parts: application, control, and data plane.

Figure 6 shows that the data plane has no flow. The control plane makes a logical decision. The data plane is responsible for transmitting and receiving packets based on commands received from the control plane. The control plane is the interface from the application plane to the data plane. The control plane can manage the information of all data planes concentratedly [15]. The series of actions means that the organization network can be managed and scalable easily. The application plane is where network developers write programs to control the network traffic. We will write a program here to check, determine if the DNS query from the clients has validity, and lose the packet if it is malicious by transmitting a drop signal to the control plane. There are many SDN APIs available on the Internet. In this paper, we create the controller program using Python based SDN solution Ryu [16] which provides support for the OpenFlow protocol.

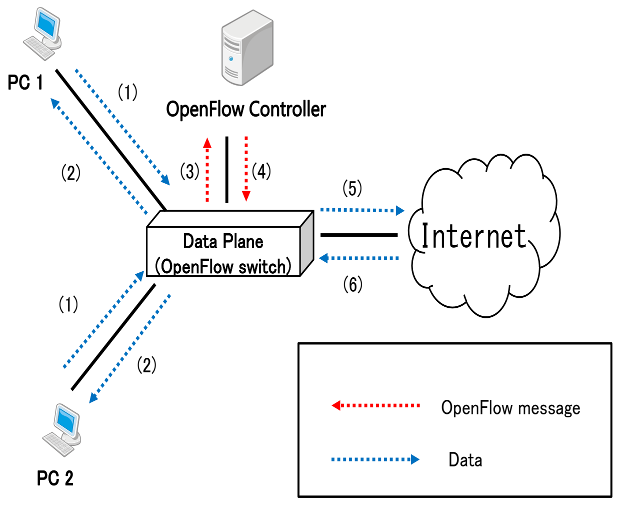

Fig. 6. Overview of OpenFlow

C. System architecture

Our proposed system architecture will use the following terminologies.

- Query Fully Qualified Domain Name (Query FQDN): The hostname or domain name that PC queries for domain name resolution.
- Query destination IP Address: The destination IP address to which PC sends the DNS queries.
- DNS RPZ: For storing the legitimate DNS NS and the corresponding A records and the MX records including legitimate public DNS servers.
- OpenFlow Switch: The network appliances used in the SDN network for transferring the packets by the flow tables created under the commands of the OpenFlow controller.
- OpenFlow Controller: The network software used in the SDN network to regulate network traffic comprehensively by commanding OpenFlow switch to create flow tables.

Figure 7 depicts a brief system architecture of our proposed system using an SDN-based network. We simply explain the process of the proposed system using a case in which a PC transmits a direct outbound application to the Internet in the following.

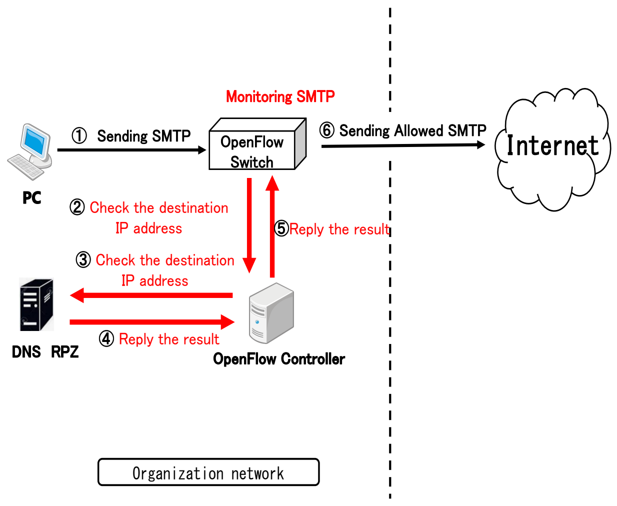

Fig. 7. System Architecture

1) In the case of using e-mail, a PC sends SMTP to the Internet and the OpenFlow switch will receive the SMTP.
2) The SMTP is received by the OpenFlow switch. Then, the OpenFlow controller is forwarded the SMTP since there is no flow table in the information about the destination IP address of the SMTP.
3) When the SMTP is received by the OpenFlow controller, the information of the SMTP is analyzed to obtain the query name and the destination IP address. The OpenFlow controller checks the destination IP address of the SMTP in the RPZ.
4) The OpenFlow controller receives the corresponding entry from the RPZ if there is the destination IP address in the RPZ.
5) The OpenFlow controller makes a decision whether or not the SMTP blocks based on the result. If the

destination IP address of the SMTP is not in the RPZ, then the OpenFlow controller blocks it as a malicious SMTP, otherwise, the OpenFlow controller will pass through the SMTP.

6) When the destination IP address of the SMTP is registered in the RPZ, the SMTP will be enabled to be transmitted to the Internet.

With these process, the direct outbound SMTP with the IP addresses stored in the RPZ by the legitimate DNS to comply with domain name resolution in an organization network will be passable as a legitimate SMTP and others will be detected and dropped as an anormal SMTP.

IV. Implementation and Evaluation

Based on the design, we constructed two programs in order to implemente the prototype system: the program to register records in the DNS RPZ and the controller program of manipulation in the OpenFlow switch. We also evaluated the prototype system in order to confirm the functionalities of the proposed system.

A. Implementation and network environment

In order to analyze the DNS traffic and register the corresponding records in the DNS RPZ, we implemented the analyzing tool for DNS traffic using a Python program [17], including the DPKT and dnspython modules. Specifically, the analyzing tool obtains the NS records, the corresponding glue A records, and the MX records, then registers the domain names with "127.0.0.1" as a legitimate A record and "127.0.0.25" as a legitimate MX record to the DNS RPZ. In addition, our implemented environment incorporated Ryu and Open vSwitch, which are OpenFlow controller and OpenFlow switch as one of the numerous SDN products for SDN-based network environment. The created Ryu program checks the destination IP address of the SMTP connection request in the DNS RPZ and determines how to handle the flow in the SDN. In the case that the destination IP address is included in the DNS RPZ, the SMTP connection request will be passed by the Ryu program, otherwise, it will be dropped.

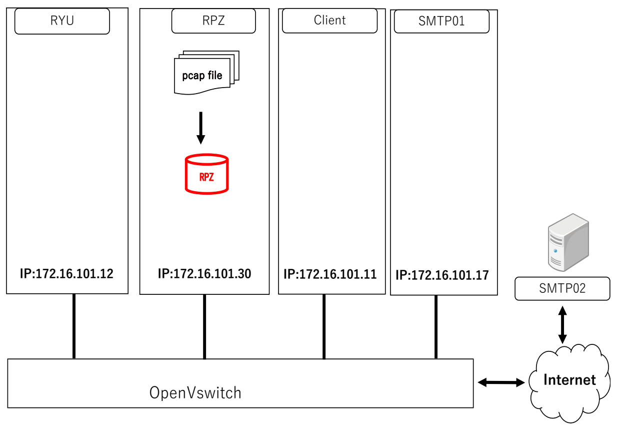

Fig. 8. Network Environment

Figure 8 illustrates the experimental network used for the feature evaluation of the prototype system. The Open vSwitch was installed on a physical machine and the other components including the Ryu controller, the client, and the SMTP01 server were installed as KVMs in the host machine. Then the SMTP02 server which is working as the external SMTP server is installed in another physical machine.

B. Feature Evaluation

We evaluated the features of the prototype system in an SDN-based network environment. Particularly, we conducted the following evaluations: measurement of DNS traffic processing time, passing and blocking DNS queries, and passing and blocking the SMTP connection request.

First, we used the DNS traffic achieved in our campus network for measuring the processing time of registering the NS, corresponding glue A, and MX records in the DNS RPZ. The analysis tool was executed for processing the PCAP file with a size of 200MB on the KVM running DNS RPZ. As a result, it took about 35 minutes to extract those DNS records and register them in the DNS RPZ.

After that, we checked the Ryu program feature for the DNS RPZ. In particular, we sent a DNS query from the client and checked the actions of the Ryu controller. We configured the DNS RPZ with the following two items for the evaluations.

1) "8.8.8.8": This IP address is one of the Google Public DNS server IP addresses [18] and the NS record and the corresponding glue A record were obtained beforehand. We registered as "8.8.8.8.rpz.example.com A 127.0.0.1" in the DNS RPZ.
2) "8.8.4.4": This IP address is another IP address of the Google Public DNS server [18] and it was not registered in the DNS RPZ.

The results of the feature evaluation for the DNS RPZ are shown in Table I. When the DNS query is recieved by the Open vSwitch for the first time, it will be transferred to the Ryu controller through the "packet_in" process since the Open vSwitch has no entries for the packets sent from the client in the flow table. After that, the Ryu program queries the DNS RPZ for the destination IP addresses such as "8.8.8.8.rpz.exmple.com A" and "8.8.4.4.rpz.example.com A" queries, and confirmes if the queries can achieve the answer with "127.0.0.1". In the evaluations, we verified that the action of Ryu controller blocked the DNS query to "8.8.4.4" correctly since it is not registered in DNS RPZ and passed the DNS query to "8.8.8.8" since it received the answer with "127.0.0.1".

At last, we checked the passing and blocking features of the SMTP server connection using the telnet command [20] from the client. As shown in Table II, the client communicated directly with an SMTP01 server(172.16.101.17) using the "telnet" command. Note that submission port 587 [19] in the SMTP01 server(172.16.101.17) was used for the consideration in actual network environment. First, the SMTP02 server (192.168.12.18) in the external network blocked the telnet from the client in the case of not storing as "192.168.12.18.rpz.example.com A 127.0.0.25" the DNS RPZ. After that, we registered

TABLE I
THE RESULTS OF FEATURE EVALUATION IN DIG COMMAND

command run on client	check from Ryu to RPZ using dnspython	check result on Ryu	Behavior of OpenFlow switch
dig @8.8.4.4 google.com mx	8.8.4.4.rpz.example.com A	null	blocked
dig @8.8.8.8 google.com mx	8.8.8.8.rpz.example.com A	127.0.0.1	pass

TABLE II
THE RESULTS OF FEATURE EVALUATION IN TELNET COMMAND

Command	Behavior of OpenFlow switch
telnet 172.16.101.17 587 (RPZ: 127.0.0.25)	pass
telnet 192.168.12.18 25 (RPZ: empty)	block
telnet 192.168.12.18 25 (RPZ: 127.0.0.25)	pass

"192.168.12.18.rpz.example.com A 127.0.0.25" in the DNS RPZ. Therefore, when we tried again, the SMTP connection request was passed through as we expected. We also confirmed that the Ryu controller worked correctly based on the DNS RPZ.

V. Discussion

We have mentioned the SMTP application in the previous section. In this section, we refer to the extension of the numerous applications based on the proposed system. DNS full resolver and authoritative name server have not only the NS, A, and MX records but also other resource records like TXT record [4], DNSKEY record [21], etc. DNS resource records severally determine the usage so that some applications need to acquire specific records such as the MX, SRV, NAPTR records, and so on. Analyzing the specific record and storing it in RPZ allow our proposed system to detect and block other direct outbound applications. Thus, our proposed system can detect and block the flexible system against abnormal applications. Furthermore, we consider the reverse direct outbound application (direct inbound application) of our proposed system. IP addresses of the organization network are also stored in the DNS RPZ beforehand. As a consequence, the source IP address of the direct inbound application is checked by the DNS RPZ. If the source IP address and the destination IP addresses are contained in the DNS RPZ, the applications can pass through. Otherwise, the application traffic will be blocked. With these designs, network administrators will be able to maintain the internal network securely for legitimate applications.

VI. Summary

The objective of this research is to have a proposition for a solution to detect and block abnormal applications by analyzing DNS traffic using DNS RPZ. We have established a virtual network environment. After that, an original system was implemented based on our proposed method by using SDN technologies. Furthermore, we made an evaluation of the feature in the experimental network environment. As a result, our proposed system showed to detect and block malicious applications correctly.

In future work, we are planning to use this RPZ in the actual network environment. Moreover, we will verify the efficiency of other abnormal applications including SIP servers, and perform performance evaluations in other large-scale network environments.

References

[1] Cofense, "2023 Cofense annual state of email security Report," https://cofense.com/wp-content/uploads/2023/03/2023-Annual-Report-Cofense.pdf, Accessed at Apr. 28, 2023.

[2] "Avast Threat Labs," https://decoded.avast.io/martinchlumecky/dirtymoe-1/, Accessed at Apr. 28, 2023.

[3] Z. Zhu, G. Lu, Y. Chen, Z.J. Fu, P. Roberts, and K. Han, "Botnet research survey," *Proc. IEEE Int'l Computer Software and Applications Conf. (COMPSAC2008),* Turku, Finland, Jul. 2008, pp. 967–972.

[4] H. Ichise, Y. Jin, and K. Iida, "Analysis of DNS TXT record usage and consideration of botnet communication detection," *IEICE Trans. Commun.*, vol. E101-B, no. 1, pp. 70–79, Jan. 2018.

[5] J. Liu, et al. "Botnet: Classification, attacks, detection, tracing, and preventive measures." *EURASIP J. Wireless Communications & Networking,* article 692654, 11 pages, Aug. 2009.

[6] A.M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, "Detection of malicious payload distribution channels in DNS," *Proc. IEEE Int'l Conf. Communications (ICC2014),* Sydney, Australia, June 2014, pp. 853–858.

[7] S.N.T. Vu, M. Stege, P.I. El-Habr, J. Bang, and N. Dragoni, "A survey on botnets: Incentives, evolution, detection and current trends," *Future Internet,* vol. 13, no. 8, article 198, July 2021.

[8] H. Ichise, Y. Jin, K. Iida, and Y. Takai, "NS record history based abnormal DNS traffic detection considering adaptive botnet communication blocking," *IPSJ J. Information Processing*, vol. 28, pp. 112–122, Feb. 2020.

[9] H. Ichise, Y. Jin, and K. Iida, "Policy-based detection and blocking system for abnormal direct outbound DNS queries using RPZ," *Proc. Int'l Conf. Future Computer & Commun. (ICFCC 2022)*, Feb. 2022, pp. 327–332.

[10] Internet Systems Consortium, "Response policy zones (RPZ)," https://www.isc.org/rpz/, Accessed on Apr. 28, 2023.

[11] C. Partridge, "Mail routing and the domain system," *IETF RFC974,* Jan. 1986.

[12] J. Rosenberg, and H. Schulzrinne, "Session initiation protocol (SIP): Locating SIP servers," *IETF RFC3264,* June 2002.

[13] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, and I. Mizukoshi, "Detecting mass-mailing worm infected hosts by mining DNS traffic data." *Proc. ACM SIGCOMM Workshop on Mining Network Data (MineNet'05),* Aug. 2005, pp. 159–164.

[14] D. Whyte, P. Oorschot, and E. Kranakis, "Addressing SMTP-based mass-mailing activity within enterprise networks," *Proc. IEEE Computer Security Applications Conf. (ACSAC'06),* Miami Beach, FL, USA, Dec. 2006, pp. 393–402.

[15] Open Networking Foundation, "SDN Architecture," https://opennetworking.org/wp-content/uploads/2014/11/TR_SDN-ARCH-1.0-Overview-12012016.04.pdf Accessed on Apr. 28, 2023.

[16] NTT, inc., "Ryu: Getting Started," https://osrg.github.io/ryu-book/en/html/, Accessed on Apr. 28, 2023.

[17] "Python," https://www.python.org, Accessed on Apr. 28, 2023.

[18] Google, "Introduction to Google public DNS," https://developers.google.com/speed/public-dns/docs/intro, Accessed on Apr. 28, 2023.

[19] R. Gellens, J. Klensin, "Message submission for mail," *IETF RFC6409*, Nov. 2011.

[20] J. Postel, and J. Reynolds, "TELNET protocol specification," *IETF RFC854*, May. 1983.

[21] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNSSEC resource records ," *IETF RFC4034*, Mar. 2005.