A study on detection and blocking of DNS-based botnet communication

by

Hikaru Ichise

A Thesis Submitted to
Graduate School of Information Science and Technology

in fulfillment of the requirements for the degree of

Doctor of Information Science and Technology

at the

HOKKAIDO UNIVERSITY

February 2020

Preface

In recent years, though the next-generation Internet provides higher QoS and various new capabilities, the number of high-impact security incidents is increasing. Some of these security incidents use malicious programs called bots, which have become a serious problem. A bot is a malicious program, which is spreading by e-mail attachments, Web sites, USB memories, etc. The bot performs various attacks such as DDoS attacks and sending spam mails. The bot-infected PC receives various attack commands from a server called the Command and Control (C&C) server, and attacks the others PCs according to the commands. Botnet communication is a logical network between a C&C server and bot-infected PCs. It is important for network administrators to detect and to remove bot-infected PCs in their organizations. In this thesis, I focus on botnet communication using DNS queries, which is a typical communication protocol used by various latest botnets, and discuss a system that automatically detects and blocks it. The thesis includes five chapters.

Chapter 1 gives the background, objective and overview of the proposed solutions in this thesis. First, I present the threats of botnet communications, and the issues of those studies, and discusses the purpose of this study. Existing researches on botnet communication have not analyzed (1) legitimate usage and unconfirmed usage of DNS TXT records, and (2) DNS traffic without using official DNS full resolver for the use of direct outbound DNS queries in botnet communication. Moreover, (3) most researchers have not been realized automatic detection and blocking of botnet communication of direct outbound DNS queries.

Chapter 2 introduces the details of the botnet, the DNS protocol, and the behavior of the botnet using DNS. I also introduce existing researches, and the issues to be solved.

In chapter 3, I discuss solutions of issues (1) and (2). In order to solve issues (1)

and (2), I first differentiate between legitimate and suspicious usages of the DNS query and then analyze real DNS query data obtained from a campus network. I discover and divide DNS queries sent out from an organization into three types — via-resolver, and indirect and direct outbound queries — and analyze the DNS query data separately. I use a 99-day dataset for via-resolver DNS TXT queries and an 87-day dataset for indirect and direct outbound queries. The results of my analysis show that about 30%, 8% and 19% of DNS queries in via-resolver, indirect and direct outbound queries, respectively, could be identified as suspicious DNS traffic. Based on my analysis, I also consider a comprehensive botnet detection system and have designed a prototype system.

In chapter 4, I study the problem (3). Namely, I design and implement the system to detect and block the botnet communication using direct outbound DNS query as an advancement from the botnet detection system roughly proposed in chapter 3. I design and implement the system to detect and block the botnet communication using direct outbound DNS query. In the proposed mechanism, all DNS traffic of an organization will be captured and analyzed in order to extract all NS records which will be stored in a white list database. Then all the outgoing DNS queries will be checked and those destined to the IP addresses that are not included in the white list will be blocked as DNS-based botnet communication. I have implemented a prototype system and evaluated the functionality in an SDN-based experimental network. The results showed that the prototype system worked well as I expected and accordingly I consider that the proposed mechanism is capable of detecting and blocking some specific types of DNS-based botnet communication.

Chapter 5 summarizes the results of this research and presents the future research directions which should be future issues.

Acknowledgments

Writing this thesis, I have received a great deal of assistance, support, and guidance from people around me. I would like to sincerely thank them all for supporting me everything throughout Doctoral course.

I would first like to express my gratitude to my supervisor, Prof. Katsuyoshi Iida, not only for his academic support and knowledge throughout my Ph.D. study and research but also for kindly giving me many amazing opportunities.

I would like to sincerely thank Prof. Yoshiaki Takai for accepting me as a part of his laboratory and suggesting many things.

Besides above, I would like to express my appreciation to the rest of my thesis committee: Prof. Hiroyuki Minami, and Prof. Masaharu Munetomo for their encouragement and invaluable comments.

My sincere thanks also goes to Prof. Yong Jin for his useful comments of my research deeply. He provided insightful comments and comprehensive corporration not only about DNS but about network security. The discussions about my research enable me with him to obtain the good research findings.

I would like to leave my thankfulness to all members in the laboratory for very enjoyable and friendly environment. There were a lot of precise times I spent together. I also thank Mr. Yuki Iuchi for advising me a lot of things in Hokkaido University and Prof. Isao Yamada for encouraging enrollment of Hokkaido University in Tokyo Institute of Technology.

I thank my all colleague in Tokyo Tech for their friendship and care. Last but not the least, I would like to put my deepest gratitude to my family who always love and stay beside me all the time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Objective in this study

Recently, the Internet is one of the important infrastructure in many kinds of organizations, and therefore cyber security protection is crucial. Example of cyber attacks include unauthorized access [1], DDoS attacks [2], spam mail [3], etc. One source of these cyber attacks is “botnet”. For example, in [4], the authors have reported that many DDoS attack incidents were caused by botnets. Moreover, in [5], the authors showed that botnets was used for spamming and spreading malware, information leakage, click fraud, and identity fraud.

Many security incidents about botnet have been reported in the literature. In [6], the authors reported that more than 500,000 computers were infected by a bot program called “MyKings” in 2018. It has created crypto currency equivalent to US $2.3 million. In [7], the authors have reported that, in 2018, a bot program called ”Viro”, which distributed spam emails containing ransomware.

In another perspective, the number of botnet activities is increasing. Figure 1.1 depicts the number of newly detected Command & Control (C&C) servers in each year between 2014 and 2018 in [8].

Here, I briefly introduce botnet. Botnet communication consists of a logical network between the bot-infected PCs and the servers sending instruction to the bot-infected PCs, which is called C&C server. PCs become bot-infected PCs by attachment files of e-mail, USB devices, and drive by download attacks. Bot-infected PCs receive illegal instructions

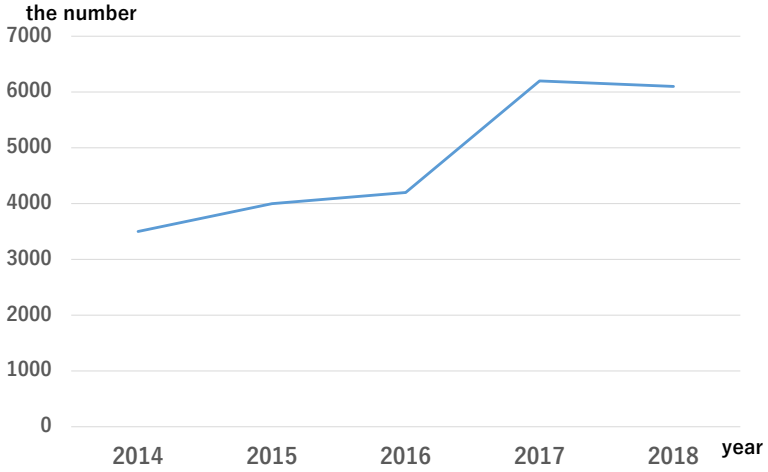

Figure 1.1: Total of newly detected Botnet C&C Listings in 2014-2018

from C&C server. Next, they perform malicious activities or attacks by obeying C&C server instructions.

Therefore, it is important for network administrators not to set bot-infected PCs in organizations. However, the latest botnet communication have used various technique. For example, in [9], the C&C in botnet communication have used domain flux and IP flux, which is the dynamic changing of domain name and IP address information, respectively. Network topology of C&C refers to four type: star, multi-server, hierarchical, and random. In addition, botnets are reported to use Domain Generation Algorithms (DGA), which generate alarge number of domain names based on an initial seed in [10]. By using DGA, botnets can search C&C easily and with stealth. In [11], examples of botnet communication protocol are of Internet Relay Chat (IRC), Hyper Text Transfer Protocol (HTTP), and Peer to Peer (P2P). The latest botnet protocol is Domain Name System (DNS) protocol. DNS is one of important network tool for the Internet users in [12]. Furthermore, DNS protocol is regarded as a safe protocol since the ability of DNS is very limited. In this reason, attackers of botnets are heavily using DNS protocol.

The focus of this thesis is to detect and block DNS-based botnet communications to maintain the security level of organization networks. As the important process, my objective is to create the system detecting and blocking the latest DNS-based botnet communication. To achieve my objective, I focus on DNS-based botnet communication in this thesis. Then, I perform two steps. In the first step, I will analyze the real DNS traffic to categorize them based on their usage. Based on the analysis, I will develop basic system

architecture of detection and blocking system of DNS-based botnet communication. In the second step, I will develop basic system architecture of detection and blocking system of DNS-based botnet communication.

1.2 Overview of this thesis

This chapter introduced the background and objective in this study. Threat of cyber security is described. The source of many attacks is botnet communication. Thus, botnet communication is one of the biggest threat. Now, botnet communication is used by DNS protocol. It is important for network administrators to detect and block the latest DNS-based botnet communication. Therefore, my objective is to create the system detecting and blocking DNS-based botnet communication.

Chapter 2 summaries an overview of botnet and DNS protocol and related work of this thesis in various points of view. First, the analytical studies are examined for DNS TXT record and botnet communication knowledge. The implementation studies of botnet detection and related technologies in terms of false positive, machine learning, and honeypot are explored. This chapter shows the gap of botnet research in which this thesis will fulfill it.

In Chapter 3, I analyze DNS traffic to develop basic detection method of botnet communication. Namely, I focus on DNS TXT record and investigate the DNS traffic obtained from the DNS full resolvers of campus network for via-resolver type botnet communication. For indirect and direct outbound queries, I investigate DNS traffic log obtained at the gateway of the campus network to the Internet. Based on the analysis, I also consider a detection system for the three types of DNS-based botnet communication and show the basic system architecture.

In Chapter 4, I design and implement the system to detect and block the botnet communication using direct outbound DNS query as an advancement from the botnet detection system roughly proposed in chapter 3. Based on the proposed method, I construct a prototype system for detecting and blocking anomaly DNS traffic. In particular, for detecting a botnet communication that uses direct outbound DNS queries, I analyze the achieved NS record and the corresponding glue A record from campus network and create

a legitimate NS record history database. I evaluate the features of the prototype system and also perform some preliminary performance evaluations using a local experimental network. According to the evaluation results, I confirm that the prototype system worked effectively as it was expectable to deploy the proposed method in real network environment.

The conclusion of Chapters 3, and 4 is put at the last part, Chapter 5. It sums up the contribution of each previous chapter and expresses eventually what the whole thesis has been done and found. Finally, the suggestion for further study of DNS-based botnet communication is leaved and opened for future.

Chapter 2

Botnet communication and related work

As I described in the Introduction, the objective of this study is to detect and block DNS-based botnet communications. In this section, I describe botnet and the communication protocols used in botnet communications, and related researches of this study.

2.1 Overview of botnet

Figure 2.1 illustrates a brief overview of botnet topology. First, some PCs are infected by malicious software in some way which is distributed by emails, web browsing, and so on. After that, the bot-infected PCs will find their master's C&C server and they also will try to communicate with their C&C server. Then, C&C server occasionally sends commands and information to the bot-infected PCs to make them attack some target hosts. Typical examples of botnet based cyber attacks include spam mail dissemination and DDoS attacks. In this way, botnet consists of a logical network between C&C server and many bot-infected PCs.

Historically, botnet has been born in about twenty years ago, and it has changed its communication protocol between C&C server and bot-infected PCs in sometimes. At the beginning, Internet Relay Chat (IRC) protocol [13] was well used in botnet communications. Then, HTTP (HyperText Transfer Protocol) and P2P (Peer to Peer) protocol based botnet communications have been developed and widely used [11, 14, 15]. And recently,

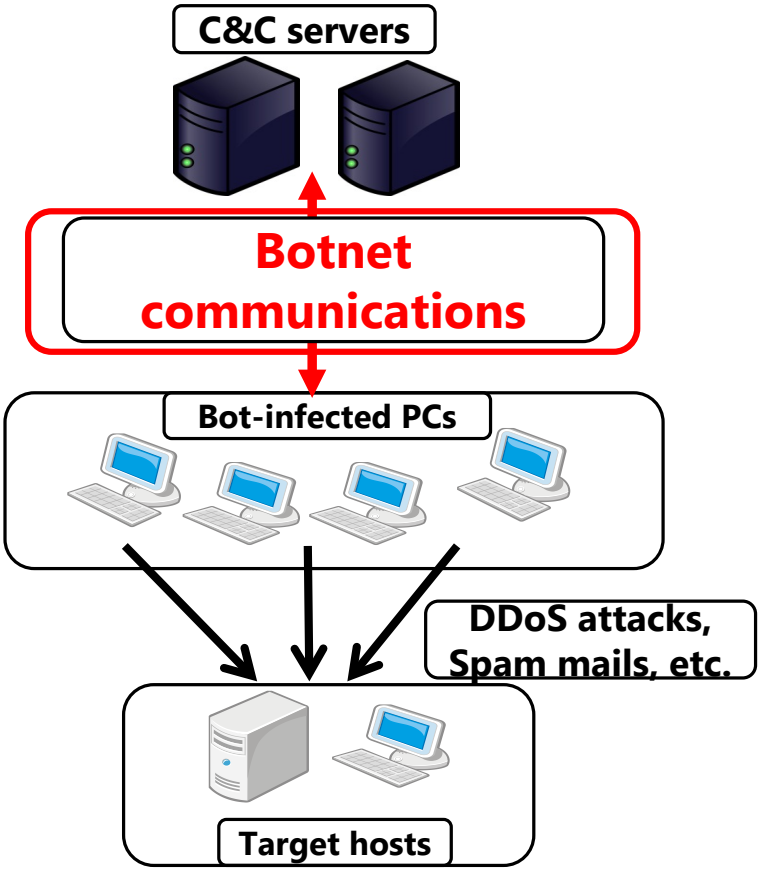

Figure 2.1: Overview of botnet communication

several independent research groups have reported the existence of DNS-based botnet communications [11, 16, 17, 18]. According to the reports, DNS-based botnet communications mainly use the TXT resource record to transport commands and controls in DNS packets.

2.2 DNS protocol and DNS-based botnet communication

Although the major objective of DNS protocol is to provide name resolution between the hostname to IP address, DNS also provides some other supplementary functions using MX, TXT and other resource record types. Among them, the TXT records provide a field to store some short text descriptions. The original RFC1035 only provides 512 Bytes for each DNS response packet including TXT resource record. Now each TXT record can store 4,096 Bytes through EDNS extensions [19]. This large space can allow us to store relatively large information including SPF [20] records and domainkeys [21] which are

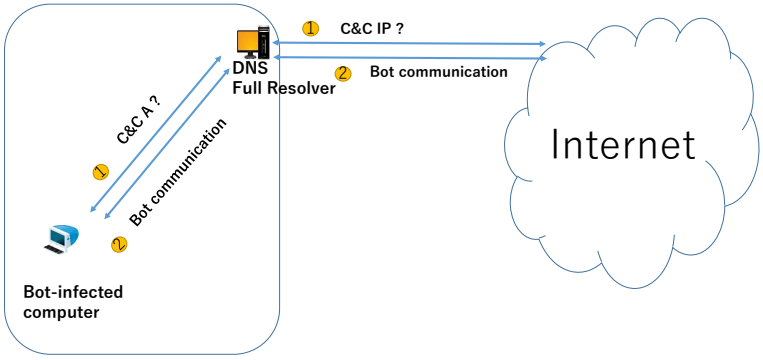

Figure 2.2: An example of via-resolver DNS query

used to deal with spam mail protections. There are also other legitimate usages of the DNS TXT resource record type exist but I omit the detail description here.

Unfortunately, the extension standard in DNS response packet size also provides possibilities for some malicious parties to use DNS TXT records to transport malicious information useful for cyber attacks depending on its flexibility. Therefore, I should establish a differentiation method between the legitimate and malicious usages of DNS traffic especially the DNS TXT resource record type which has been appeared in botnet communications.

In general, most organizations set up multiple DNS full resolvers to provide name resolution service to their internal computers. In this case, the internal computers send name resolution requests to one of the DNS full resolvers, which performs the name resolution on their behalf. I define such a name resolution request as a via-resolver DNS query in this research and Fig. 2.2 shows an example of via-resolver name resolution. The via-resolver name resolution relies completely on DNS full resolvers, so I need to monitor all DNS traffic on the DNS full resolvers in order to analyze a via-resolver DNS query. Several researchers have reported the usage of a via-resolver DNS TXT query in botnet communication [11, 16], so the via-resolver DNS TXT query is one of my monitoring and analysis targets when I design a botnet communication detection system.

Even though official DNS full resolvers are available, some internal computers may use their private DNS full resolvers or public DNS full resolvers (from the Internet) for purposes such as name resolution and independent configuration of a DNS full resolver. In this research, I define this kind of name resolution request without using official DNS full resolvers as a direct outbound DNS query. Several studies have also reported the use

of direct outbound DNS queries in botnet communication [11, 18, 22].

Two types of outbound DNS query, which include indirect outbound query and direct outbound query, are involved in botnet communication. One is where the bot-infected computer uses the DNS full resolvers to query the IP address of the C&C server only and then sends DNS queries to the C&C server directly, as shown in Fig. 2.3(a). I refer to this as an indirect outbound DNS query. Note that the difference between via-resolver DNS query and indirect outbound DNS query is the way of bot communication, each of which is illustrated as arrow numbered two in Figs. 2 and 3(a), respectively. The second type is where the bot-infected computer never uses the DNS full resolvers, but instead sends DNS queries to the C&C server directly from the start, as shown in Fig. 2.3(b). I refer to this as a direct outbound DNS query. In this case, the IP address of the C&C server might have been embedded in the bot program at the installation stage, so the bot-infected computer can communicate with it directly without looking for its IP address via the DNS full resolvers.

As well as the DNS TXT record type, there are other exceptions where a direct outbound DNS query is a legitimate usage, such as updates of anti-virus software, the use of DNS black lists to check for spam mail, and the use of public DNS resolvers. Some anti-virus software use the DNS TXT record type to check the update status and some spam-mail checkers use the DNS TXT record type to check the domain name in DNS black lists. These usages of the DNS TXT record type are legitimate, so I need to differentiate them from malicious usages. On the other hand, several public DNS full resolvers have been launched to provide an effective name resolution service by using direct outbound DNS queries. The Google public DNS [23] is a popular public DNS resolver, which is announced with the IP addresses “8.8.8.8” and “8.8.4.4”, and some internal computers in many organizations may use these addresses for name resolution. Obviously, the DNS queries to these public DNS resolvers should be categorized as direct outbound DNS queries since they do not use the DNS full resolvers. Therefore, I also need to filter them out from malicious direct outbound DNS queries, but I have to include them into the analysis of DNS TXT usages.

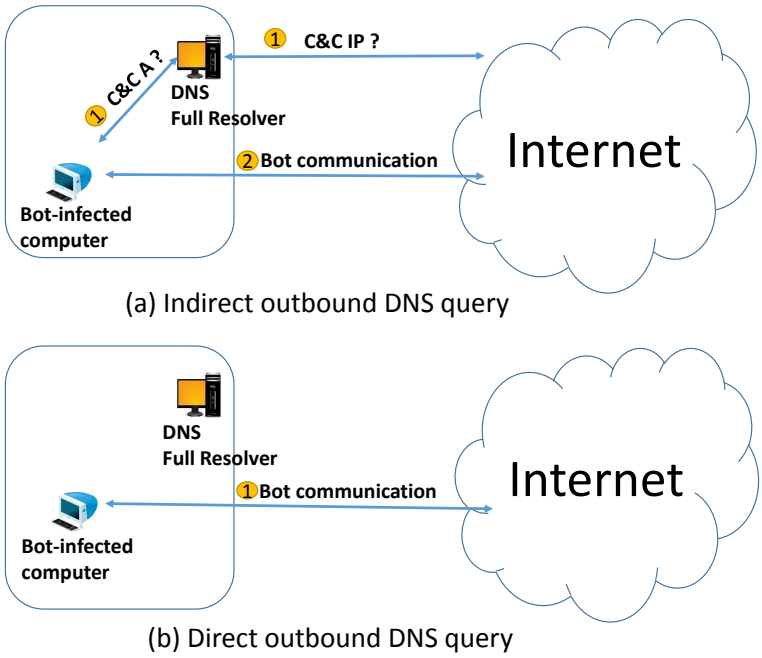

Figure 2.3: Direct & Indirect outbound DNS queries

2.3 Related work

Hiding information in DNS traffic is not a new technology. The first implementation, called DNS tunneling, appeared in 1998 [11] and the target information embedded in FQDN and CNAME was used. After that, in 2004, Kaminsky presented his implementation to tunnel arbitrary data over DNS traffic to the security community [24]. A few years later, DNS-based VPN was created [25].

There are some conventional ways to detect "abnormal" DNS traffic. In 2011, several reports discussed the existence of DNS-based botnet communication and detection approaches [18, 22, 26]. So far as I know, [26] is the earliest one about DNS-based botnet communication. It discussed ways to detect DNS abnormal usage attempts and also provided recommendations to mitigate exposure. In [22], another botnet named Morto has been reported, where DNS TXT record was also used in botnet communication. In Morto, a bot-infected computer works in collaboration with Domain Flux [27] to find C&C servers by querying the IP address of the generated domain names using DNS full resolvers. After identifying the C&C servers, the bot-infected computer performs botnet communication using DNS TXT record with them directly. In [16], the authors analyzed an archived malicious dataset covering one year and a 30-day real DNS traffic obtained

from a DNS full resolver, and they confirmed that a high ratio of DNS TXT record transactions might increase the risk of infection by botnet.

In addition, in recent years, many securities related types of the research reported that the the DNS protocol had been widely used for botnet communication as well as malware attacks [18, 28, 29, 30]. and I introduce some DNS based related solutions in the following.

In [28], the authors surveyed and categorized the existing researches about DNS-based botnet detection. The major topic of this thesis is to introduce the detection method using a DNS mechanism, it also explains some existing researches about botnets that use the DNS protocol.

In [18], the authors analyzed 14 million DNS TXT queries and found the bot program named "Feederbot" which has to query the C&C server directly, that is to say, bypassing the DNS full resolver. However, none of the existing researches considered the role of DNS NS (Name Server) records in the name resolution process. Moreover, as all reports never discussed direct/indirect outbound DNS queries so far, it was difficult to detect and block only DNS-based botnet communications. Thus, I tend to create automatic detecting and blocking system for DNS-based botnet communications by constructing the legitimate NS records history database for monitoring all DNS queries.

In [29], the authors referred to DNS tunneling technology in which the malicious contents are included in the labels of domain names being used as a part of DNS queries. To detect malicious DNS tunneling traffic, they discussed two separated categories, payload analysis, and traffic analysis. In payload analysis, they analyzed tunnel indicators from the payload of a DNS query and response packets with focusing on Domain Generation Algorithms. In traffic analysis, they analyzed over time in DNS traffic.

In [30], the authors proposed a real-time detection method of tunneling of data over DNS using machine learning techniques. The authors also implemented and preliminarily evaluated the proposed system. Since this method is based on machine learning, the performance will heavily depend on the learning datasets.

As I can see from the existing research, although the usage of DNS protocol and DNS TXT record in botnet communication has not gone unexamined, no clear conclusions regarding the official and malicious usages of DNS TXT record or the architecture of

DNS-based botnet communication (via-resolver, indirect and direct outbound) have been provided. Since DNS TXT record is also used for legitimate purposes, such as Sender Policy Framework (SPF) [20] and domainkeys [21], which are used for email sender authentication, the proper usages of DNS TXT record need to be identified in order to detect botnet communication. More importantly, in addition to the traffic of DNS TXT record, which is a medium for botnet communication, the botnet communication architecture also needs to be considered since new DNS resource records can also appear as new media. Furthermore, as I can see from the above DNS based related researches, many of them only focused on DNS traffic analysis and failed to consider the DNS name resolution process. In this thesis, I purpose to solve this issue in conventional solutions.

2.4 Conclusion

The study of botnet commuinication can be divided to two strategies. The first one is analytical research which is done by analyzing real DNS traffic for consideration of detcting and blocking DNS-based botnet communication. The other one develops the system and does experiment in the virtual environment. Both can shows the effectiveness of the system to detect and block DNS-based botnet communication.

The analytical work usually is considered DNS traffic via DNS full resolver. Many studies focused on the statistical analysis, the domain analysis and reverse engineering for detecting DNS-based botnet communication. To detect DNS-based botnet communication, the concrete analysis focus on the length of domainname and reverse engineering of real bot program and the ratio of DNS TXT record. However, the study is still lack of the usage DNS TXT recxord and direct outbound DNS queries consideration. It always based on the ideal DNS query.

The research about implementation of detecting DNS-based botnet communication concentrated on the machine learning for detecting DNS-based botnet communication. The studies proposed several possible strategies for implementation of detecting DNS-based botnet communication. As a result, Some of research show the ratio of detection. However, they did not related the direct outbound DNS query. Moreover, the detection system is very expensive for local area network. There is still a simple implementation

research.

The related work of botnet communication is widely done in many aspect to fulfill the high ratio of detection but it is difficult to detect and block DNS-based botnet communication simply for new type bot program. This thesis brings the analysis of botnet communication and simple detection method toward the prototype implementation in the virtual network.

Chapter 3

DNS traffic analysis and design of protection system against botnet communication

3.1 Introduction

Botnet, a malicious logical network of cyber attackers, has become a significant security threat in cyberspace [31, 32]. There are many well known botnets, such as Conficker, Storm, TDL4, and Zeus, each of which once infected millions of computers [33]. Once a computer is infected by a bot program, which is a kind of malware and also a core program of a botnet, it can attempt several kinds of cyber attack such as Advanced Persistent Threat (APT) , Distributed Denial of Service (DDoS), spreading spam mails, and phishing [33, 34]. Fig. 3.1 shows a typical workflow of a botnet-based cyber attack. First, a computer within an organization somehow gets infected by a bot program such as through web browsing, spam mail, or clicking a phishing site by mistake. After that, the bot program sends probes to its corresponding Command and Control (C&C) server to identify its existence as well as to update its status. After it is finished collecting a number of bot-infected computers, the C&C server can instruct them to perform several kinds of cyber attack. Here, I refer to the communication between a bot-infected computer and the C&C server, which is the most important information transmission in a botnet-based cyber attack, as *botnet communication*. With regard to the above workflow, in this research

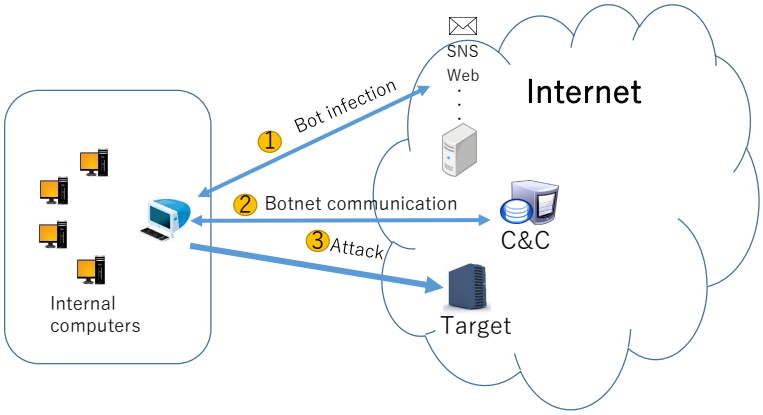

Figure 3.1: Typical workflow in a botnet attack

I target the botnet communication as a means to analyze and detect botnet-based cyber attacks.

To date, Internet Relay Chat (IRC), Hypertext Transfer Protocol (HTTP) and Peer-to-Peer (P2P) protocols have been used in botnet communication [11]. In addition, many recent reports indicate that the Domain Name System (DNS) protocol [35, 36] is also used in botnet communication [26, 37, 38]. Basically, DNS protocol is mainly used for name resolution, which is translating the hostname to an IP address in the Internet, but the increasing popularity of Internet services has led to some minor records, such as DNS TXT, which is used in many Internet services. In [17], Xu et al. empirically show that cyber attackers can effectively hide botnet communication by using a DNS-based stealthy messaging system that uses hash functions to encode the contents. In [39], Anagnostopoulos et al. show how mobile botnets use DNS protocol in botnet communication and also evaluate the magnitude of DNS-based amplification attacks. Consequently, DNS packets, which are currently considered to be secure network traffic, have also become a target of monitored communication since network administrators cannot simply block all DNS traffic. Thus, an effective detection solution for botnet communication is needed.

In this research, as a preliminary step to develop a method for detecting DNS-based botnet communication, I analyze real DNS traffic from my campus network. There are three possible ways to use DNS traffic in botnet communication: through via-resolver, indirect outbound and direct outbound communication. The via-resolver type completely uses DNS full resolvers in botnet communication which is a typical usage in normal name resolution. The indirect outbound type partially uses DNS full resolvers only to obtain the

IP addresses of C&C servers. After that, the bot-infected computers communicate with the C&C servers directly using DNS protocol. The direct outbound type does not use DNS full resolvers at all, but communicates with the C&C servers directly from the beginning. All three types have recently been detected in DNS-based botnet communication and reported in the literature [18, 22]. More importantly, recent detections also indicate that DNS TXT record, which has flexible usages compare to other DNS resource records, is increasingly being used in botnet communication.

Considering the above, in my analysis I focus on DNS TXT record and investigate the DNS traffic obtained from the DNS full resolvers of my campus network over a period of 99 days for via-resolver type botnet communication. For the other two types, I investigate all the DNS traffic, excluding the via-resolver DNS traffic, obtained at the gateway of the campus network to the Internet for 87 days. Based on my analysis, I also consider a detection system for the three types of DNS-based botnet communication and show the basic system architecture.

3.2 Analysis of via-resolver DNS TXT query

As explained in Sect. 2.2, the DNS TXT record type is used for botnet communication in various botnets. To detect botnet communication based on DNS TXT record type, legitimate usages of DNS TXT record type must be distinguished from suspicious ones. In this section, I describe my analysis of via-resolver DNS TXT query data obtained from the DNS full resolvers of my campus network.

3.2.1 DNS traffic capturing and analytical methodology

Figure 3.2 depicts a simplified network topology of the DNS traffic capture from the DNS full resolvers set up in my campus network and the methodology that enabled us to obtain the DNS traffic. As shown in the topology, I only need to capture the DNS traffic launched by the DNS full resolvers (steps 2 and 3) and only store DNS TXT query type in the DNS traffic server. With such a network configuration, I captured and stored the DNS TXT query data for 99 days (from March 24, 2014 to June 30, 2014). Note that before the analysis I made the source IP addresses anonymous to respect the privacy of individual

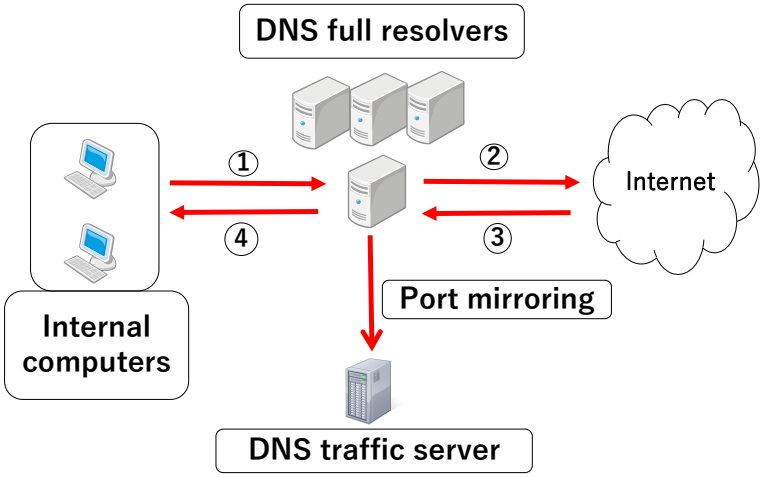

Figure 3.2: Architecture to obtain via-resolver DNS TXT query data

users. After I passively obtained the DNS TXT query data from the DNS full resolvers, I analyzed them through a two-step process:

1. Differentiate legitimate usages of the DNS TXT record type and filter out unconfirmed usages (Sect. 3.2.2)

2. Investigate the detection ratio calculated from the unconfirmed usages of DNS TXT record using a conventional method (Sect. 3.2.3)

3.2.2 Classify usages of the DNS TXT record type

To identify legitimate usages of the DNS TXT record type and filter out unconfirmed usages, I statistically analyzed the obtained DNS traffic according to all published official usages of the DNS TXT record type. Table 3.1 lists some official usages of the DNS TXT record type as well as unconfirmed categories. These are limited to some typical examples I have found, so it is possible that new usages will appear with the introduction of new applications. Considering these possible usages of the DNS TXT record type specifically, I performed the following procedures to categorize the obtained DNS TXT query data.

- SPF (RFC4408) and DomainKeys (RFC4870): For this category, I filtered the responses for DNS TXT query which included “v=spf” and “domainkey” strings.

- DNS-based Service Discovery (RFC6763): For this category, I identified the DNS TXT queries which included “._dns-sd” in the FQDNs.

Table 3.1: Usages of DNS TXT record

Category	Basis	Usages
SPF	RFC7208	Formatted regular TXT record
DomainKeys	RFC6376	Formatted regular TXT record
DNSSD	RFC6763	Formatted regular TXT record
NFSv4	RFC 7530	-
Anti-virus	e.g., Sophos	Base64 encoded long TXT record
Spam and DNSBL	-	-
P2P tracker	-	
NTP	RFC5905	-
Misc	Various	Various long TXT record
Unconfirmed	Various	Base64 encoded long TXT record

Table 3.2: Usages and statistics of DNS TXT record

Category	# of queries	Ratio [%]
SPF and domainkey	12,223	0.24
DNS-based service discovery	213,978	4.30
NFSv4	3,596,481	72.14
Anti-Virus	597,901	12.00
SPAM Check and DNS Blacklist	180,600	3.63
P2P Tracker	446	0.01
NTP	632	0.01
Misc	380,723	7.63
Unconfirmed	2,293	0.04
Total	4,985,277	100

- NFSv4 (RFC3530): I identified and filtered out the DNS TXT queries which included "_nfsv4idmapdomain" in the FQDNs.

- Anti-Virus software: This category includes the update processes of anti-virus software which use the DNS TXT record type. I identified DNS TXT queries that included ".sophos." [40], "immunet" [41], and AVG corporation's domain [42] in the FQDNs.

- Spam email check and DNS black lists: This category includes specific domain names for spam email check and DNS black lists. I identified the DNS TXT queries

which included “spamcop.net” [43], “spamhaus.org” [44], “rbl.maps.vix.com” and “sa-accredit.habeas.com” in the FQDNs.

- P2P tracker: This category represents P2P trackers used by BitTorrent. I identified the DNS TXT queries included “bttorent” and “tracker” strings in the FQDNs.

- NTP (RFC1305): Some corporations use the DNS TXT record type to obtain the IP addresses of NTP servers. I identified the DNS TXT queries which include “ntp minpool” and “time” in the FQDNs.

- Misc.: This category includes miscellaneous applications and campus internal communications. For miscellaneous applications, I identified the DNS TXT queries which include: “time.asia.apple.com”, “apple.com” (push notifications for mail deliveries by Apple), “planex.co.jp” (software updates for network devices manufactured by Planex Corporation), “gateway.com”, and “xmpp.org” [45] in the FQDNs. For internal communication, I identified the DNS TXT queries which included “titech.ac.jp” in the FQDNs.

- Unconfirmed: This is a group of usages of the DNS TXT record type which were not included in any of the above categories.

The last category, “Unconfirmed”, contains instances that cannot be added into any other category. Those usages of the DNS TXT record type have not been announced officially by specific application vendors nor are they based on any standard protocols based on DNS. Therefore, these “Unconfirmed” usages possibly include suspicious information exchange, such as DNS-based botnet communication. The statistical results are shown in Table 3.2. Note that I only counted the DNS TXT queries that received responses since bot-infected computers need to exchange information with the C&C servers. Here, I call a query-and-response pair as a query having a response. Consequently, in my statistical analysis I obtained 2,293 “Unconfirmed” DNS TXT query-and-response pairs as suspicious.

Table 3.3: “Virustotal.com” check results

	DNS TXT queries	unique IP addresses	IP detection	URL detection	IP and URL	IP or URL	Ratio of IP or URL
Total	4,985,277	2,334	161	190	123	228	9.77% (228/2334)
Unconfirmed	2,293	330	55	87	42	100	30.3% (100/330)
Unconfirmed / Total	0.0460%	14.1%	-	-	-	43.9%	-

3.2.3 Results of via-resolver DNS TXT query analysis

In Sect. 3.2.2, I investigated official usages, announced legitimate usages as well as unconfirmed usages of the DNS TXT record type and obtained 2,293 “Unconfirmed” usage of DNS TXT query from approximately five million query-and-response pairs of DNS TXT record. This means we can greatly reduce the number of query-and-response pairs required to be investigated from five millions to 2,293 through extracting only the unconfirmed usage. The next question is how is the detection ratio through such extractions of the unconfirmed usage. In this section, I therefore investigate the detection ratio calculated by the conventional method. The results are described in Table 3.3.

Before calculating the detection ratio, I first count the number of destination IP addresses in the DNS TXT queries. The number of total DNS TXT queries is of 4,985,277, which corresponds to 2,334 unique destination IP addresses, whereas that of unconfirmed TXT queries is of 2,293, which corresponds to 330 unique destination IP addresses, as shown in the column of “# of unique IP addresses” in Table 3.3. This equivalents that extractions of the unconfirmed usage reduced the number of destination IP addresses to 14.1%. However, if the extracted results do not include botnet communications, my method is not effective. To investigate the effectiveness, I use “Virustotal.com” [46], a free third-party security check web site. “Virustotal.com” provides a brief check of target IP addresses to see if they are involved in downloading suspicious files and hosting URLs to identify malware, infected files, malicious web sites, etc. Among them, “Searching for URL scan reports” and “Searching for IP address information” can be used for evaluation of my research. The former method, I call “URL detection” in this research, is based on the database of malicious URLs provided by URL scanners. And the latter method, I call “IP address detection” in this research, is based on the IP address database provided by antivirus solutions. Both detection methods can detect suspicious communications from the list of IP addresses. The experiment of “Virustotal.com” was performed at Dec. 12,

2016. [1]

Table 3.3 shows the detection results of "Virustotal.com" by comparing with the "Unconfirmed" usages. From the results, first, I need to point out that the total detection ratio of my analysis (30.3%) is much higher than "Total" usages that do not use my analysis (9.77%). That is, among the 2,334 IP addresses in total, the "Virustotal.com" detected 228 IP addresses in URL or IP detection; and among the 330 IP addresses in Unconfirmed usages, 100 IP addresses were matched with the URL or IP detection. Note that "URL or IP detection" means the sum of "URL detection" and "IP address detection". This will give us the detection results with the widest coverage of suspicious IP addresses among detection methods I used. Details on each detection category can be referred to Table 3.3. Next, note that the cost effectiveness of the detection is much higher in my analysis. That is, in my analysis, I detected about 43.9% (100/228) of IP addresses which might involve in malicious communication only by investigating about 14.1% (330/2334) of that in the conventional method. Note that my method cannot detect 128 (= 228 − 100) IP addresses. This is because Virustotal.com is not specialized to detect only bot communications; there exist suspicious IP addresses not by bot communications. To detect such IP addresses, I have to use other methods by collaboration with my proposed method.

In all, in my analysis I extracted 330 unique destination IP addresses of DNS TXT record queries that might have been involved in malicious communication during the 99 days, which means an average of 3.3 IP addresses per day. Considering there are four DNS full resolvers set in my campus network and based on the network traffic conditions of my university, three or four times as many unique destination IP addresses will be detected as suspicious per day (about 13 IP addresses) from an organization with the same scale as that of my university. I consider this to be a reasonable number for handling by human operators.

[1] The results described in [47, 48] are different from this thesis. This is because the experiment in [47, 48] was performed at Apr. 18, 2015, which is different date with this thesis. Even if the same list of IP addresses sent to Virustotal.com, the statistical results will be changed if I tested at different date.

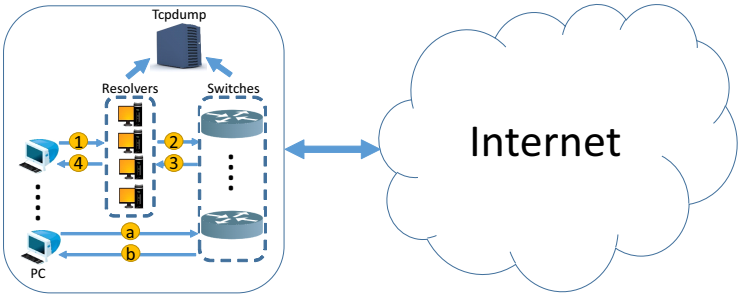

Figure 3.3: System topology for DNS traffic capture

3.3 Indirect/dirct outbound DNS TXT query analysis

As described in Sect. 2.2, indirect and direct outbound DNS TXT queries are used for botnet communication in various botnets. To detect botnet communication using indirect and direct outbound DNS TXT queries, it is important to distinguish between legitimate and suspicious usages of the DNS TXT record in indirect and direct outbound DNS TXT queries. In this section, I describe my analysis of indirect and direct outbound DNS TXT query data obtained from my campus network. Note that the query data I analyze in this section do not include the via-resolver DNS TXT query data.

3.3.1 DNS traffic capturing and analytical methodology

I obtained real DNS traffic from my campus network over a period of about three months (87 days) and analyzed all indirect and direct outbound DNS TXT query data. The system topology I used to obtain the DNS traffic is shown in Fig. 3.3. There are four DNS full resolvers set up in my campus network and I assume that some users use them while others send DNS queries to the outside DNS servers in the Internet directly. Thus, I captured all DNS traffic in the border routers and investigated the possibility of DNS TXT record usage in botnet communication. To respect the privacy of internal users, I made the IP address of the internal computers in the captured DNS packets anonymous in a one-to-one manner in order to trace the behavior in the corresponding name resolution (steps 1, 4, a and b in Fig. 3.3) after they had been made anonymous (except 2 and 3 due to they belong to the official full DNS resolvers) I captured the DNS traffic in my campus network from Nov. 1, 2014, to Jan. 26, 2015, and analyzed the DNS TXT query data through the following steps.

1. Capture all DNS traffic by mirroring from the border routers and filter out valid query-response pairs. Here, "valid" means that in a query-response pair the query is initialized from an internal computer and the corresponding response is returned.

2. Filter out NS records from the query-response pairs obtained in step 1 and map corresponding glue A records that can also be obtained from the same pairs or successive queries for the glue A record in case of out-of-bailiwick NS record [49]. After that, store them in a database called NS_DB.

3. Capture all DNS TXT queries sent directly from the internal computer to the outside and obtain their query destination IP addresses, and then check if NS_DB created in step 2 has these addresses.

4. If the destination IP address of a direct outbound DNS TXT query is in NS_DB, go to check the next record; otherwise, log it for further investigation.

5. Based on the check results of step 3, I create two unique destination IP address lists: one is for indirect outbound DNS TXT queries and the other is for direct outbound DNS TXT queries.

6. To confirm the results of my analysis, I also checked the IP addresses in the two IP address lists through a security check site, "Virustotal.com".

In general, I believe that any direct outbound DNS query can be involved in some kind of malicious communication and it depends on the convenience of the DNS resource record type. Since use of the DNS TXT record type in botnet communication has been reported, however, I only analyzed DNS TXT record types that were involved in direct outbound DNS queries.

3.3.2 Results of indirect/direct DNS TXT query analysis

The number of unique destination IP addresses captured per day in indirect and direct outbound DNS TXT queries and the corresponding results from the "Virustotal.com" check are shown in Figs. 3.4 and 3.5, respectively. As shown in Fig. 3.4, the total number of unique destination IP addresses captured in direct outbound DNS TXT queries varies

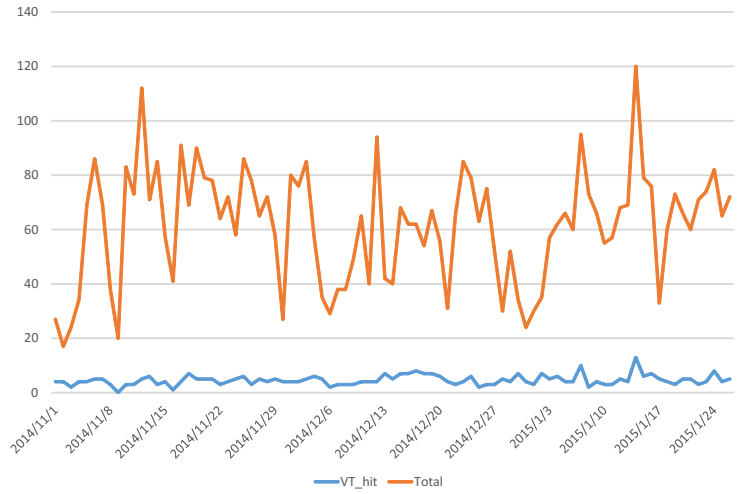

Figure 3.4: VirusTotal results for direct outbound DNS query

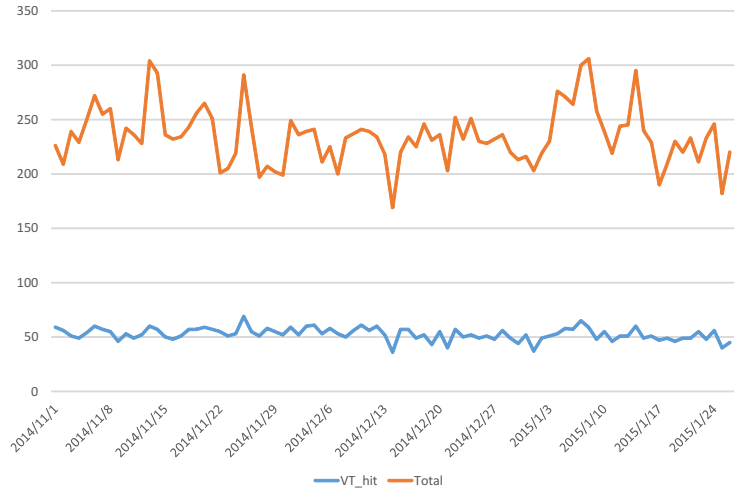

Figure 3.5: VirusTotal results for indirect outbound DNS query

approximately from 15 to 120, and the average number of unique IP addresses per day is 62. On the other hand, as shown in Fig. 3.5, the total number of unique destination IP addresses in indirect outbound DNS TXT queries varies approximately from 160 to 305 and the average number of unique IP addresses per day is 235. These numbers may include duplicated IP addresses, though, since I only divided captured DNS traffic on a daily basis without considering the TTL-based cache in this analysis. Therefore, in actual operation the number of IP addresses that I would need to process is less than 300 per day, which is a reasonable number that will become smaller if I can remove the duplicated IP addresses.

Next, I checked the destination IP addresses captured from the indirect and direct

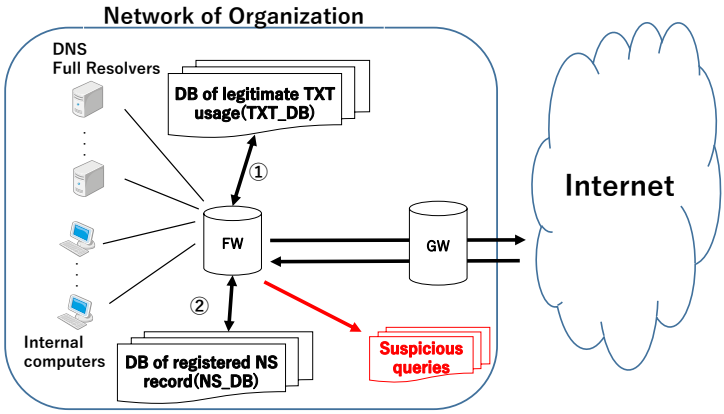

Figure 3.6: Architecture of a DNS-based botnet detection system (covers via-resolver, and indirect and indirect outbound DNS TXT queries)

outbound DNS TXT queries on "Virustotal.com". As Fig. 3.4 shows, the hit rate of the IP addresses captured in direct outbound DNS TXT queries is comparatively stable at about 5 per day (the average hit rate is about 8%). This number includes the IP addresses that saw hits in URL-based detection or IP-based detection or both. This result indicates that for direct outbound DNS queries, I need to investigate in detail about 5 IP addresses per day, which seems not to be large for the network administrators. On the other hand, the results for the indirect outbound DNS TXT queries are not so good, as I can see from Fig. 3.5. The average hit number of IP addresses per day reached about 53 (the average hit rate is about 22%), which means about 19% IP addresses in total had hits in "Virustotal.com" check. I also checked the destination IP address lists to see which public DNS resolvers were used in the campus network and found only Google public DNS resolvers were involved; i.e., 8.8.8.8 and 8.8.4.4. Since public DNS resolvers can also be used by bot-infected computers to search for the C&C servers belonging to indirect outbound DNS TXT queries, I have to include these as part of a target. Regular name resolution must retrieve the corresponding authoritative NS records and their IP addresses during its process. However, the destination IP address of a direct outbound DNS query cannot be traced in NS_DB, and in an indirect outbound DNS query it is not usual to use the DNS full resolver partially in one single name resolution process. Finally, in DNS-based botnet communication, even if an infected computer repeatedly sends a direct outbound DNS query to the same IP address (a C&C server), a method with NS_DB can successfully detect it since there is no valid NS record corresponding to the destination IP

address in NS_DB.

My results indicate that it is possible that the destination IP addresses captured in indirect and direct outbound DNS TXT queries may have been infected by some kind of malware, especially those in direct outbound DNS queries. This is because direct outbound DNS query requires hard coding of IP addresses for servers, which seem to be suspicious compared with indirect outbound one. Therefore, it is appropriate to detect botnet attacks in the early stage (while searching for C&C server and performing botnet communication) by monitoring indirect and direct outbound DNS TXT queries. Note that DNS protocol is continuously used in botnet communication after C&C server has been identified because it is easy to avoid being monitored. In this evaluation, I used "Virustotal.com" for further investigation and the main purpose of that was to confirm the effectiveness of my analytical method. Thus, in actual operation I can take action based just on the monitoring results. Considering over blocking and misdetection, I believe it can be mitigated by prerequisite leaning for legitimate IP address of name servers. In addition, like other security solutions, my analysis also has a zero-day vulnerability [50], but this is a challenge with respect to all anti-virus and security-related solutions. Therefore, I need to be very careful when creating policies for the destination IP addresses obtained in the indirect and direct outbound DNS TXT queries.

3.4 Consideration of Botnet Communication Detection

In Sect. 3.2, I investigated DNS TXT record type to find via-resolver-based botnet communications. In Sect. 3.3, I examined botnet communication based on either indirect or direct outbound DNS TXT queries. In this section, I discuss the possibility of a comprehensive DNS-based botnet detection system that would cover all the three query types (via-resolver and indirect/direct outbound DNS query) and look at the design of such a system. Via-resolver and indirect/direct outbound DNS queries based bot communications are different. I do not tend to compare the detection methods of via-resolver DNS query and indirect/direct DNS query. I tend to categorize legitimate usages of DNS TXT record type in via-resolver DNS queries and block the malicious destination IP addresses detected in indirect/direct outbound DNS queries. This system is based on botnet com-

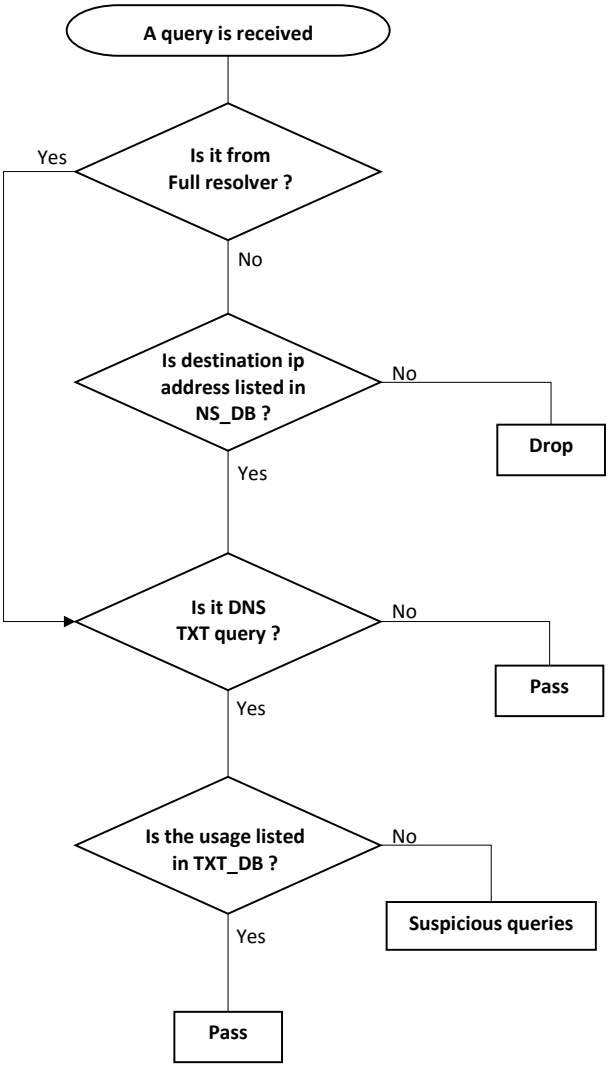

Figure 3.7: Flow chart of comprehensive detection system

munication detection by monitoring DNS TXT queries. The basic idea of the system is as follows.

1. Via-resolver DNS TXT queries need to be checked to confirm legitimate usages.

2. All received NS (Name Server) information, including its FQDN and glue A record, needed in an organization must be stored in a database and updated periodically.

3. The destination IP addresses of all indirect and direct outbound DNS queries must be checked if they are included in the database.

Based on the above points, I introduce two databases into the system, as shown in Fig. 8. The first one (TXT_DB) is for storing information regarding legitimate usages of

the DNS TXT record. The second one (NS_DB) is for registering valid NS information. These two databases are used to filter unconfirmed DNS TXT queries and check indirect/direct outbound DNS TXT queries which can probably be used for detecting botnet communication.

The detection procedure of the comprehensive detection system works as shown in Fig. 9, which is a flow chart of the system. When a firewall (denoted by FW in the figure) detects a DNS packet, it first checks the source IP address of the packet. If the source IP address belongs to an official DNS full resolver and the record type is DNS TXT record, the system checks the usages of the DNS TXT record type in the TXT_DB. If the usage of the DNS TXT record type is listed in the TXT_DB then the DNS packet will be passed. On the other hand, if the usage of the DNS TXT query is not listed on the TXT_DB, information regarding the target DNS TXT query will be logged for further investigation.

Next, I consider the other case, when the source IP address does not belong to an official DNS full resolver, which is in the top branch of the flow chart. In this case, the system first checks whether the destination IP address of the DNS query is listed on NS_DB. Since direct outbound DNS queries without using an official DNS full resolver are unusual, I must drop the query if it is not listed on NS_DB. If the destination IP address of the DNS query is listed on NS_DB and the query is the DNS TXT record type, the system then checks TXT_DB. If the DNS TXT query is not listed on TXT_DB, I should log a suspicious queries entry similar to that for the DNS full resolver case. Note that for the special destination IP addresses such as official public DNS resolver, I will register them in the NS_DB in advance.

After the system has collected "suspicious queries," the network operators should investigate the IP addresses listed on the suspicious queries with collaboration of other security facilities to confirm the fact. Basically, I consider the collected queries are suspicious due to they use abnormal usages of DNS protocol. However, this botnet detection system also has some shortcomings. First, it can only detect botnet communication using the DNS TXT record type, which means I cannot find botnet communications using the A and/or CNAME record type. Second, the attacker may implement a DNS TXT-based application that is very similar to a "legitimate" usage, and this will make it more difficult to identify suspicious DNS traffic. Third, the system needs collaboration with

other published security information and I may not able to detect highly suspicious botnet communication if there is no necessary information.

To overcome the first issue, I can extend the botnet detection system to support A and/or CNAME record types. For the second one, I need to find deeper characteristics of "true legitimate applications" that cannot be mimicked by attackers. For the last one, I need to develop an advanced method to find "highly suspicious" botnet communication — e.g., by using multiple third-party security check web sites in combination to ensure the necessary data is always available. Finally, I also need to consider about name resolution privacy. DNS over Transport Layer Security (TLS) [51] has been defined as a standard and I need to add a new feature to obtain DNS queries under its deployment. These solutions will be part of my future work.

3.5 Conclusion

Although the use of DNS TXT record type in botnet communication has been widely reported, the literature does not provide a comprehensive botnet detection system to prevent such botnet communication. My goal in this research has been to detect three types of botnet communication based on DNS TXT queries: through via-resolver as well as indirect and direct outbound DNS TXT queries.To deal with the via-resolver DNS TXT queries, I analyzed 99-day DNS traffic of TXT records obtained from DNS full resolvers of my campus network and categorized its usages. To deal with indirect and direct outbound DNS TXT queries, I analyzed 87-day DNS traffic of TXT records obtained from the border gateway of my campus network, which did not include the via-resolver DNS TXT query data.

In the first case, I significantly reduced the number of IP addresses needed to be investigated in detail (from 2,334 to 330) by extracting "Unconfirmed" usages of DNS TXT record. I then confirmed that among the "Unconfirmed" DNS TXT queries about 30.3% were identified as "highly suspicious" and it had about 43.9 % common detection rate with a conventional security check site "Virustotal.com.". In the latter case, through a similar analysis, I found that about 19% and 8% of destination IP addresses were identified as "highly suspicious" in indirect and direct outbound DNS TXT queries, respectively.

Based on my analysis, I discussed the possibility of a comprehensive botnet detection system and proposed a design for such a system. Although this botnet detection system has shortcomings, I intend to find ways to overcome these in my future work.

Chapter 4

Implementation and evaluation of protection system against botnet communication

4.1 Introduction

Botnet, a malicious logical network constructed by cyber attackers, has become one of the critical security threats in cyberspace [31, 32]. Once a computer is infected by a bot program, which is a kind of malware and also a core program of a botnet, the bot-infected computer basically attempts several kinds of cyber attacks such as Advanced Persistent Threat (APT), Distributed Denial of Service (DDoS), spreading spam mails, ransomware attacks, phishing [33, 52], etc. In general, the botnet-based cyber attack can be divided into infection, botnet communication and attack. First, a computer in an intranet, which is an internal computer network within an organization such as universities, companies and governmental departments, etc., somehow gets infected by a bot program such as through web browsing, spam mail, or clicking a phishing site by mistake. Then, the bot program sends probes to its corresponding Command and Control (C&C) server to identify its existence as well as to update its status. After collecting a number of bot-infected computers, the C&C server can instruct them to perform several kinds of cyber attacks. Here, I refer to the communication between a bot-infected computer and the C&C server, which is the most important information transmission in a botnet-based cyber

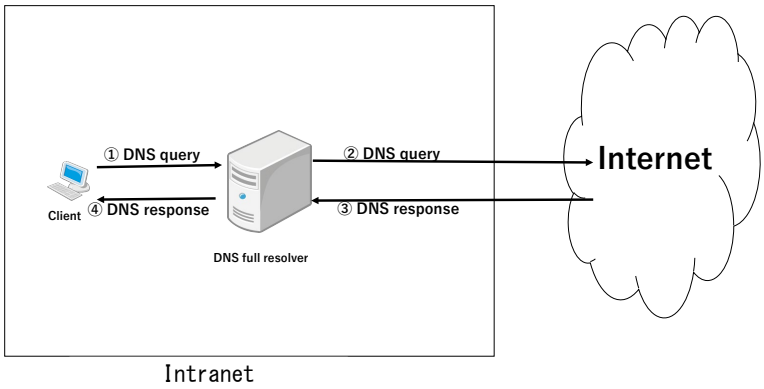

Figure 4.1: An example of normal DNS name resolution

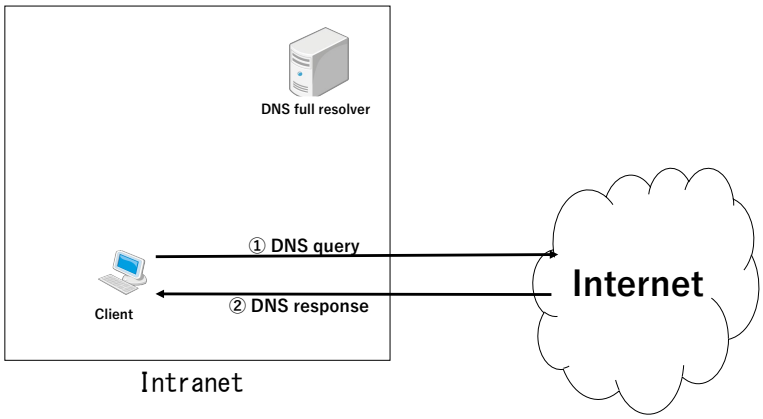

Figure 4.2: An example of abnormal DNS name resolution

attack, as botnet communication. With regarding the above workflow, in this research I target botnet communication as a means to analyze and detect botnet-based cyber attacks.

Many recent reports indicate that DNS protocol [35, 36] has become used in botnet communication [26, 37, 38]. DNS protocol has been mainly used for name resolution, such as translating hostnames to IP addresses on the Internet. However, the increase of Internet services has led to wide uses of some minor records such as DNS TXT record. In [17], Xu et al. empirically showed that cyber attackers can effectively hide botnet communication by using a DNS-based stealthy messaging system that uses hash functions to encode the contents. In [47, 53], Ichise et al. analyzed DNS packet traces to differentiate the normal and abnormal uses of DNS TXT records. Consequently, DNS traffic which so far has been considered to be secure network traffic has also become a target of being monitored communication since network administrators cannot simply block all DNS traffic. Thus, it is important for the network administrator to detect and block DNS-based botnet communications.

Figure 4.1 shows a general DNS based name resolution process with a deployed DNS full resolver in an intranet. A client computer first sends a DNS query to the DNS full resolver for requesting name resolution. Then the DNS full resolver performs the name resolution and replies back the DNS response to the client computer. In the name resolution process, the DNS full resolver achieves the corresponding DNS NS (Name Server) records and glue A records of the authoritative DNS servers prior to sending DNS queries to them. Here, a NS record is used for indicating the hostname of an authoritative name server of the queried domain name, and its corresponding glue A record means the IP addresses of the authoritative name servers. In almost of all cases, the internal clients rely the DNS name resolution on the DNS full resolvers deployed in the intranet.

On the other hand, botnet communications may not follow the same process [18, 29] as I will explain in the next section. Figure 4.2 shows a typical anomalous DNS traffic such as a type of botnet communication using DNS protocol. In this example, a client computer sends a DNS query to the Internet directly without using the DNS full resolver deployed in the intranet. I call this type of DNS name resolution request as a direct outbound DNS query. However, there also exist some exceptions that this type of direct outbound DNS Q&R can be used for normal use cases. For example, in the case of using public DNS servers such as Public DNS operated by Google [54] and/or Cloudflare [55], a client computer may send direct outbound DNS queries to the public DNS servers without using the DNS full resolvers deployed in the intranet. Therefore, in the proposed system, I also consider these exceptions and allow the computers in the intranet use the public DNS servers.

In summary, DNS name resolutions are supposed to be performed by DNS full resolvers in almost all normal cases, whereas it is not general in abnormal use cases. Based on this observation, I propose a detection and blocking method of anomalous DNS traffic. My key idea is that normal name resolution obtains the NS and glue A records of the corresponding authoritative DNS servers prior to sending DNS queries to them while bot programs send direct outbound DNS queries without obtaining the NS and glue A records.

In this thesis, I focus on detecting and blocking anomalous DNS traffic by analyzing the achieved NS records history. Based on my proposed method, I constructed a proto-

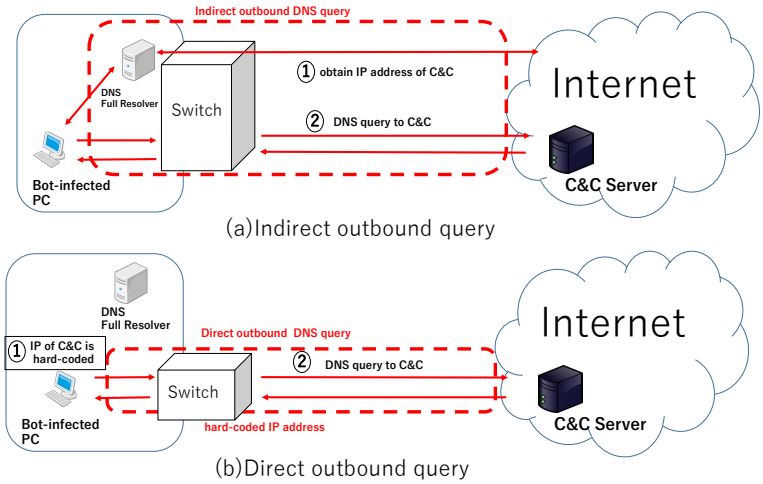

Figure 4.3: Direct & indirect outbound DNS queries in

type system for detecting and blocking anomaly DNS traffic using virtual machines. In particular, for detecting a botnet communication that uses direct outbound DNS queries, I analyze the achieved NS record and the corresponding glue A record from my campus network and created a legitimate NS record history database. I evaluated the features of the prototype system and also performed some preliminary performance evaluations using a local experimental network. According to the evaluation results, I confirmed that the prototype system worked effectively as it was expectable to deploy my proposed method in real network environment.

4.2 DNS-based botnet communication

As stated in the Introduction, the objective of my research is to construct a system for detecting and blocking DNS-based botnet communication. In this section, I introduce three types of DNS-based botnet communication; the way of using via-resolver DNS query, the way of using direct outbound DNS query and the way of using indirect outbound DNS query.

4.2.1 Three types of DNS-based botnet communication

I find out that there are three types of botnet communication using DNS; via-resolver DNS query, indirect outbound DNS query, direct outbound DNS query. The way of using

via-resolver DNS query relies on DNS full resolver completely. In [11], the authors have reported the uses of DNS TXT records in botnet communication using via-resolver DNS query. Next, botnet communication using indirect outbound DNS queries is detected in a bot program named Morto [22], which uses direct outbound DNS queries after identifying its C&C server. Figure 4.3(a) shows that a bot-infected computer obtains the IP address of C&C servers by name resolutions via DNS full resolver at first, then sends DNS queries to a C&C server directly. On the other hand, a bot program named Feederbot never uses DNS full resolvers, which is called a direct outbound DNS query. Figure 4.3(b) shows that the IP address of C&C servers are hard-coded in the bot program so that the bot-infected computer can send DNS queries to C&C server using the IP address directly.

These two types of DNS-based botnet communication which are used in Morto and Feederbot, never obtain legitimate NS records and the corresponding glue A records before sending DNS queries to the C&C servers. In this thesis, I define "legitimate NS record", "normal DNS query and response" and "abnormal DNS query and response" as follows:

- Legitimate NS record: NS records obtained from authoritative DNS servers.
- Normal DNS query and response (Q&R): DNS queries sent to DNS full resolver or authoritative DNS servers. DNS responses received from DNS full resolver or authoritative DNS servers.
- Abnormal DNS query and response (Q&R): DNS queries sent to other than DNS full resolver and authoritative DNS servers. DNS responses received from other than DNS full resolver and authoritative DNS servers.

Fig 4.4 illustrates DNS NS record resolution process in detail. First, the client sends a DNS query to the DNS full resolver. DNS full resolver then obtains NS record of all authoritative name severs from the root server. Finally, the DNS full resolver obtains the destination IP address of "www.example.com" and notifies the destination IP address to the client.

It should be noted that DNS full resolvers and authoritative DNS servers can also be compromised and reply wrong DNS responses. In this thesis, I keep the point on the

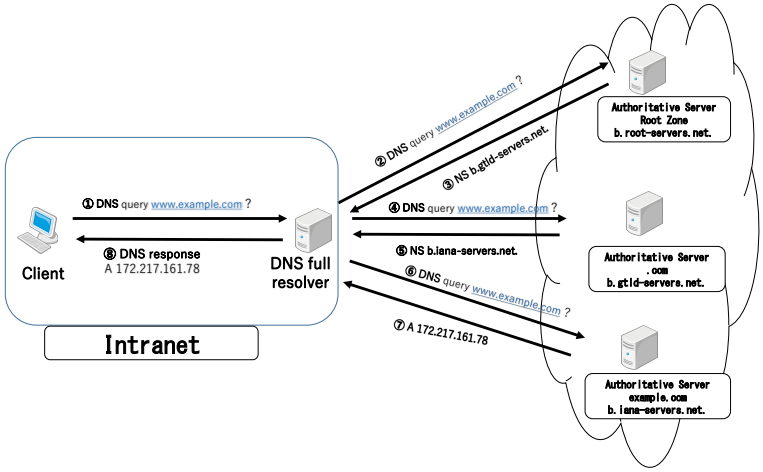

Figure 4.4: NS record process

normal DNS name resolution process by focusing on obtaining legitimate NS records prior to sending DNS queries. Therefore, the case of a compromised DNS full resolver and authoritative DNS servers is beyond the scope of this thesis and I omit the detailed discussion. Accordingly, I consider that if I can collect obtained legitimate NS records as well as the corresponding glue A records and check the destination IP addresses of all the DNS queries in the intranet, it will be possible to detect and block these types of DNS-based botnet communication.

4.3 Proposed system

In order to detect and block DNS-based botnet communications, I first construct a database of whitelisted DNS NS and glue A records. This list includes the DNS NS records of domain names achieved from all received normal DNS responses such as domain names ".com" and ".net" and their corresponding glue A records. I also include other well-known IP addresses for specific applications such as update servers of anti-virus software and public DNS servers to which the internal computers may send DNS queries directly. When a client sends direct outbound DNS queries to the listed servers I consider them as normal, while to those not listed in the white list I will drop the packets and log them down for further investigations. I use Software Defined Network (SDN) technology, specifically OpenFlow switch and controller, for detecting and blocking the DNS traffic in the proposed system. When a DNS query packet is "packet in" from the switch to the controller,

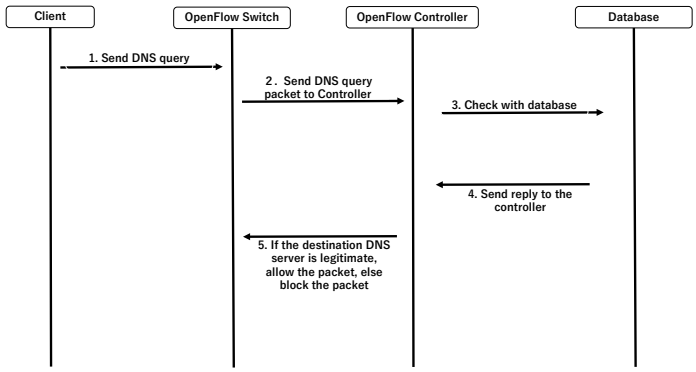

Figure 4.5: Overview of the workflow in the proposed system

the controller queries the aforementioned white list database for the destination IP address. The controller will then send instructions to the switch to drop or allow the DNS query packet. Figure 4.5 shows the brief workflow of the proposed system.

Table 4.1: Configuration of MariaDB

querytime	queryid	queryname	res_time	zname	nsttl	nsfqdn	glueAttl
decimal(20,0)	int	varchar(10000)	decimal(20,0)	varchar(1000)	decimal(20,0)	varchar(1000)	decimal(20,0)

glueA	del_time
varchar(1000)	decimal(20,0)

4.3.1 Design of NS record history database

Fig. 4.6 illustrates the detailed flow chart of NS_DB operations. First, I construct the NS_DB from the corresponding DNS queries and responses. One of the simple methods is to obtain all DNS traffic from an intranet and pick out legitimate DNS NS records as well as the corresponding glue A records. For the simplicity, I create query_DB first for storing all normal DNS queries. Next, I construct the NS_DB using all the achieved normal DNS responses to the queries stored in the query_DB. I captured all DNS traffic in pcap format using tcpdump program and analyzed them in order to construct the NS_DB. Specifically, in the pcap file, if the line is a DNS query the analysis program performs the procedure for a DNS query otherwise for a DNS response. In the DNS query procedure, a new entry will be added to query_DB while in the DNS response procedure a new entry will be added to NS_DB. Note that some responses have NS records with the corresponding glue A records while some others have NS records only. In case of where the DNS responses

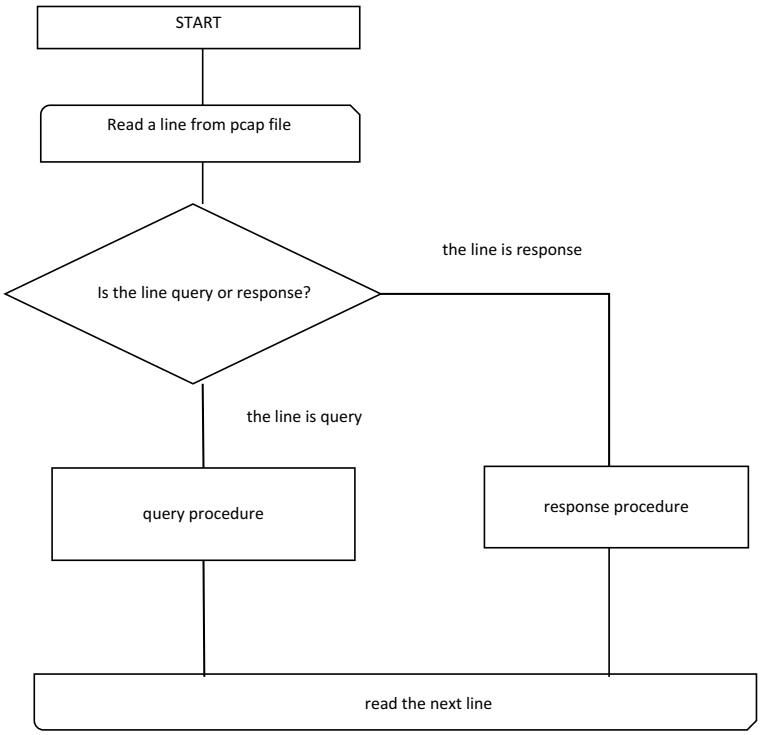

Figure 4.6: The workflow of NS_DB creation program

only have NS records (such as when an out-of-bailiwick domain name is used for NS record), only the NS records will be registered to the NS_DB. Then the analysis program continuously monitoring DNS traffic. If DNS queries for the NS records are sent and the corresponding responses are received, the glue A records obtained from the responses will be registered as glue A records in NS_DB. Thus, I defined glue A record as glue A record in an additional section and A record for out-bailiwick NS record in an answer section. I set the following column of NS_DB.

- querytime: Received time of DNS query in milliseconds
- queryid: Message id of the query
- queryname: FQDN and record type of the query
- res_time: Received time of DNS response in milliseconds
- zname: Zone name of authoritative name server
- nsttl: TTL of NS record
- nsfqdn: FQDN of authoritative name server

- glueAttl: TTL of glue A record
- del_time: Expected expire time in milliseconds calculated through nsttl or glueAttl

Table 4.1 showed as configuration of MariaDB.

4.3.2 Overview of SDN and Ryu

SDN technology [56, 57] refers to a new approach for network programmability by providing the capacity to initialize, control, change and manage network behavior dynamically via open interfaces. SDN introduces the abstraction of the data forwarding plane from the control plane. What this means is that SDN allows a network engineer to develop a program that can control, inspect and/or modify the flow of all network traffic. With this ability, I can check if the packets are sent from the client is a DNS query and how to handle them. SDN is able to perform this task by decoupling the standard routine of a switch or router into 3 separate planes: application, control, and data plane.

The lowest level, the data plane, has no logic. Logical decisions are made in the control plane. The data plane is in charge of sending and receiving frames based on instructions received from the control plane. The middle level, the control plane, is a black box that provides an interface from the application plane to the data plane. The key takeaway is that the control plane's logic can be logically centralized. This makes the management and scalability of an intranet easy. The highest level, the application plane, is where network engineers write a program to control the network traffic. I will write a program here to inspect, determine if the client's query has legitimacy and drops the packet if it is not normal by sending a dropped signal to the control plane. There are many SDN APIs available out on the Internet such as Floodlight [58]. In this thesis, I write the controller program using Python [59] based SDN solution Ryu [60] which supports the OpenFlow protocol.

4.3.3 System architecture

I use the following terminologies in the description of the proposed system:

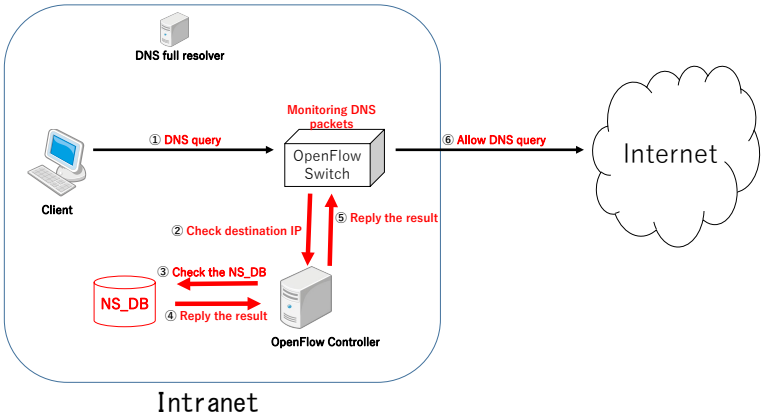

Figure 4.7: The detailed procedure of the proposed system

- Query destination IP Address: The destination IP address to which the client sends a DNS query.
- Database: For storing the legitimate DNS NS and glue A records as well as normal public DNS servers.

Basically, I use SDN technologies for controlling the packets in the proposed system. All packets will be transferred to the OpenFlow switch and the OpenFlow controller decides whether or not pass through the specific packets. In the proposed system, the destination IP address of a packet will be checked in the NS record history database. Figure 4.7 shows a simple system architecture of the proposed method by using SDN technologies. I describe the basic procedure of the proposed system using an example in which a client sends a direct outbound DNS query to the Internet in the following.

1. A client sends a DNS query to the Internet directly and the packet will be transferred to the OpenFlow switch.
2. The OpenFlow switch receives the DNS query and forwards the packet to the OpenFlow controller since the information about the destination IP address of the DNS query packet is not in the flow table.
3. When the OpenFlow controller receives the DNS query packet, it inspects the packet to obtain the queryname and the destination IP address. The controller checks the destination IP address of the DNS query packet in the NS record history database.

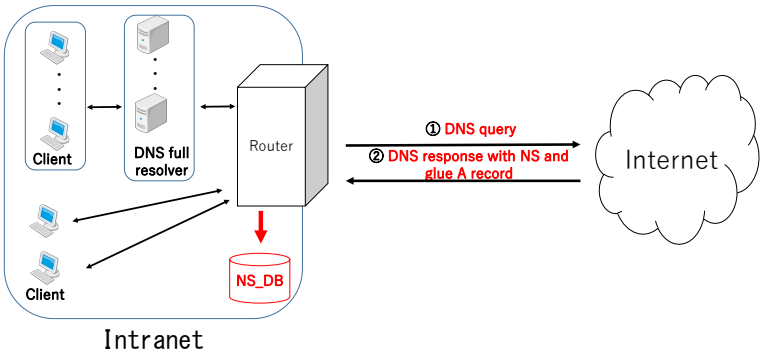

Figure 4.8: DNS data collection in case of DNS response with NS and glue A record

4. The database replies the corresponding entry (can be multiple entries) if the destination IP address is in the database.

5. The OpenFlow controller decides whether or not the DNS query packet passes through based on the result. If the destination IP address of the DNS query packet is in the database, then the OpenFlow controller passes it as a normal DNS query, otherwise, the OpenFlow controller will drop the DNS query packet.

6. In the case where the destination IP address of the DNS query packet is in the database, the DNS query packet will be allowed to be transferred to the Internet.

Based on the above procedure, the proposed system can detect and block abnormal DNS traffic which is supposed to be used in some types of DNS-based botnet communication.

Table 4.2: An example of a table entry in the NS record history database

querytime	queryid	queryname	res_time	zname	nsttl	nsfqdn	glueAttl
1414782044594	31812	yahoo.co.uk	1414782044659	yahoo.co.uk	57954000	ns3.yahoo.com	51169000

glueA	del_time
203.84.221.53	1414833213659

4.4 Implementation

Based on my proposed system, I created two programs for the implementation of the prototype system: the program to create NS_DB and the controller program to control the OpenFlow switch. I describe the detailed contents in the following sections.

Table 4.3: Software versions used in the implementation

Operating system, software and database	Version
CentOS	7.4
Open vSwitch	2.9
Ryu	4.23
Python	2.7
MariaDB	10.3

Table 4.4: The results of feature evaluation

Command	Result of checking the destination IP address by NS_DB	Behavior of OpenFlow switch
nslookup www.google.com 8.8.8.8	8.8.8.8 is unknown	blocked
nslookup www.google.com 8.8.4.4	8.8.4.4 is registered glueA record	passed
nslookup www.yahoo.co.uk 203.84.221.53	203.84.221.53 is registered glueA record	passed

4.4.1 Construction of NS record history database

In order to create the NS record history database using DNS traffic in pcap format, I created a DNS query database named query_DB first and eventually created the database NS_DB. I used Python language in the program and imported the dpkt module for processing the pcap files, and finally I used MariaDB [61] as the database system.

Figures 4.8 and 4.9 illustrate the brief network topology I used to obtain DNS traffic in my campus network. All DNS queries (the arrows numbered 1 in Fig. 4.8) and DNS responses (the arrows numbered 2 in Fig. 4.8) are obtained from the border switch in pcap format. Then the analysis program analyzes the pcap file and stores the legitimate NS records and corresponding glue A records in the NS_DB. Note that I do not construct an empty database in the initial NS_DB, but register destination IP addresses of root server and TLD in the initial setup. Table 4.2 shows an example of the entry in the NS_DB. Moreover, if the DNS response only includes NS records, that is, the corresponding glue

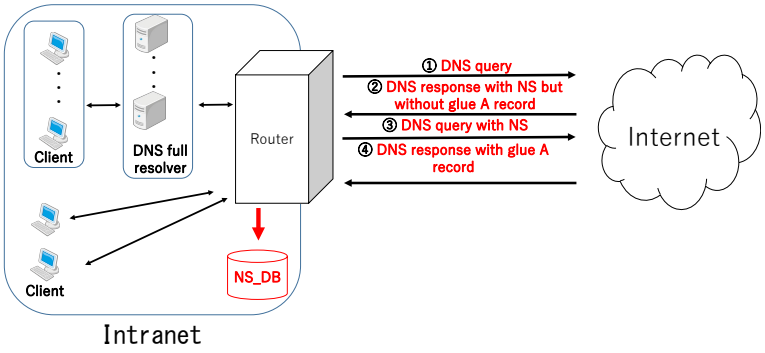

Figure 4.9: DNS data collection in case of DNS response with NS but without glue A record

A records are not included, the NS records will be registered without glue A records as Fig. 4.9 shows. After that, if the glue A records will be received continuously within two seconds the remaining information about the corresponding glue A records will be added in the NS_DB. Otherwise, the NS records will be deleted. Some destination IP addresses of the direct outbound queries are normal. For example, I need to register IP addresses of public DNS, of DNS full resolver, and of antivirus to glue A records, because those IP addresses are normal. Note that the DNS traffic of fixed interval (appropriately one month) is obtained, analyzed by setting in period of learning. Then, my system sets NS_DB in a real SDN network, obtains the DNS traffic, which update NS_DB, and runs a blocking direct outbound DNS query. The NS record and the corresponding glue A record is deleted in expiring time to live. The size of NS_DB can be restrained.

Table 4.5: The results of query time performance using dig

Command	Type of DNS resolver	Conventional method		Proposed method	
		Average [ms]	Standard deviation [ms]	Average [ms]	S. D. [ms]
dig @172.16.101.20 www.google.com (w/o cache)	Internal	515.5	92.13	530.7	69.42
dig @172.16.101.20 www.google.com (with cache)	Internal	0	0	0	0
dig @8.8.8.8 www.google.com	Public DNS (8.8.8.8)	40.9	2.33	42.9	22.85
dig @1.1.1.1 www.google.com	Public DNS (1.1.1.1)	4.6	2.00	4.0	1.25

4.4.2 Detection and blocking features

In the prototype system, I used the SDN technology to realize the detection and blocking features, specifically OpenFlow solutions. I choose Open vSwitch [62] and the OpenFlow switch and Ryu program as the OpenFlow controller. The former is a software version of the OpenFlow switch program and the latter is a solution of OpenFlow controller program. I constructed the network environment using Kernel Virtual Machine (KVM) and used CentOS 7.4 operating system for both host and guest machines. Table 4.3 shows the detailed version information for the operating system, SDN solutions and database system. In the prototype system, all packets will be transferred to the Open vSwitch and the Open vSwitch only checks DNS packets. When the Open vSwitch cannot find an entry for a DNS query packet, the "packet_in" function will send the DNS query packet to the Ryu controller. Then the Ryu controller checks the destination IP address of the DNS query packet in the NS_DB in order to determine how to process it. If the destination IP address is stored as a glue A record in the NS_DB, the DNS query packet will be passed

Table 4.6: The results of query speed performance using dnsperf

Command	Type of DNS Resolver	Conventional method			Proposed method		
		Average query speed [query/s]	S. D. of q. s. [q/s]	Limitation of max. q. s. [q/s]	Average q. s. [q/s]	S. D. of q. s. [q/s]	Limitation of m. q. s. [q/s]
dnsperf -s 172.16.101.20 domainlist (w/o cache)	Internal	80.13	5.81	108	40.14	0.23	42
dnsperf -s 172.16.101.20 domainlist (with cache)	Internal	2976.59	409.52	N.A.	2955.29	1014.65	N.A.

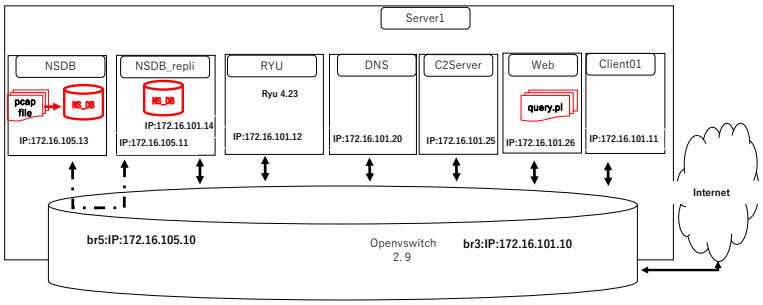

Figure 4.10: The network topology used in evaluation

through the Open vSwitch by the Ryu program. Otherwise, the DNS query packet will be blocked by the Ryu program. It should be noted that in the current system other traffic will be passed through the Open vSwitch.

4.5 Evaluation

In order to verify the feature and performance of the proposed system, I evaluated my system in a local testbed. In the feature evaluation, I mainly focused on the functionality of the controller program to check the ability of the system about detecting and blocking malicious DNS traffic. The evaluations include database construction part and malicious DNS traffic detection and blocking part. Next, to quantitatively evaluate the performance, I conduct the stress test by comparing the conventional and proposed systems.

4.5.1 Experimental network environment

To conduct the evaluation of the prototype system, I have set up a local experimental network environment consisting of an OpenFlow swtich (Open vSwitch), an OpenFlow controller (Ryu), a NS record history database (NS_DB) which is created from pcap files, a replica database to be accessed from OpenFlow controller, a client, C2Server to send a direct outbound DNS query, Web to download program with a direct outbound DNS

query, and a DNS, which DNS application is BIND version 9.9.4. These are shown in Fig. 4.10. The OpenFlow switch was installed on a host OS, whereas the controller (RYU), client (Client01), DNS (DNS), replica database (NSDB_replica), C2Server, Web, and NS record history database (NSDB) were installed as Kernel Virtual Machine (KVM) on host OSs. The controller, DNS, client, C2Server, Web, and replica database were configured to be connected to the same network in an OpenFlow switch and they could only communicate with the global Internet through the OpenFlow switch. The replica database and NS history database were connected to another network in the OpenFlow switch and synchronized by Galera Cluster, which is one of the most advanced software for multi-master synchronous replication [63]. Moreover, NS_DB can use index for improving search performance. I used NSDB_replica in order to avoid a deadlock caused by simultaneous update and reference for the same entry. That is NSDB_master will be used for maintaining the database while the NSDB_replica will be used for search and reference. More specifically, I created a replica database from the NS record history database using Galera Cluster. Note that records of the NS record history database will be deleted when del_time expires. Consequently, all traffic from the client and the DNS will pass through the OpenFlow switch and thus the OpenFlow controller can check the destination IP addresses as well as the result of packet filtering at the switch. In my previous study [66], I obtained many pcap files of DNS traffic from the border router of my campus network. In the evaluation, I used the pcap files for constructing the NS record history database. A pcap file of DNS traffic for about two hours approximately has a size of 200MB and I used it in the evaluation. It took one hour to analyze the pcap file and to store the NS records history in the database. In particular, from the pcap formatted file, I matched the DNS queries and the responses. Next, I registered all NS records in the authority section and the corresponding glue A records in the additional section to the NS_DB. If the response only has NS records without glue A records like out-of-bailiwick domain name, I only registered the NS records in the NS_DB. Note that the normal destination IP addresses, such as Public DNS, were registered to NS_DB. As a result, 18,100 legitimate NS records were picked from the pcap file and stored in the NS record history database.

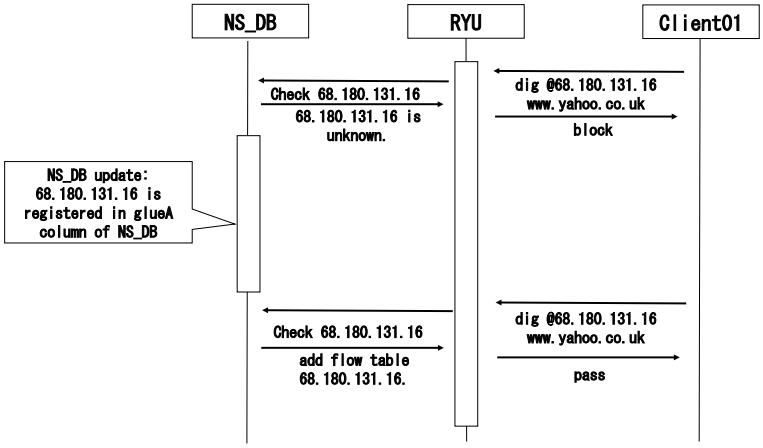

Figure 4.11: Check registration of glue A in NS_DB

4.5.2 Feature evaluation

After constructing the NS records history database in the experimental network environment, I evaluated the malicious NS traffic detection and blocking feature of the proposed system, which is mainly implemented in the controller program. Specifically, I attempt to check the destination IP address of a DNS query packet sent from the client with the glue A records stored in the NS record history database (NS_DB) and the OpenFlow switch will be instructed by the controller program to block the DNS query if there was no glue A record in the database.

To test the controller program, I sent DNS query from the client (172.16.101.11) through the command described in Tab. 4.4. When the OpenFlow switch receives a DNS query packet having a new query request, "packet_in" packet will be forwarded to the controller program, which will check whether the destination IP address is stored in the database. If the destination IP address is stored as a glue A record in the NS_DB, the controller program will create a log entry "<destination IP address>is registered as a glue A record" and the query has to be passed and passes the DNS query packet. Otherwise, it will log "<destination IP address>is unknown" and the query has to be blocked and blocks the DNS query packet.

I described the evaluation results in Tab. 4.4. I tested for three types of IP addresses as the destination IP address of the DNS query from the client; 8.8.8.8 which is an open DNS resolver provided by Google and I did not register it in NS_DB to be able to perform the test, 8.8.4.4 which is another open DNS resolver provided by Google and is registered

in NS_DB, and 203.84.221.53 which is an authoritative DNS server of the domain name "yahoo.co.uk" and is registered in the NS_DB.

In addition, I checked the feature of the program to manipulate the NS_DB with the flow illustrated in Fig. 4.11 in the following. Firstly, I made the NS_DB empty and then made the client send the DNS query using "dig @68.180.131.16 www.google.co.uk" to the Internet with running the RYU controller. The results showed that the RYU controller blocked the DNS query with "68.180.131.16 is unknown" message due to the destination IP address "68.180.131.16" does not exist in NS_DB. Next, I updated the NS_DB with adding the IP address "68.180.131.16" as a glue A record which eventually showed "68.180.131.16 is registered in glue A column of NS_DB" message. I checked that there is "68.180.131.16". Then I also manually checked the NS_DB and confirmed the IP address "68.180.131.16" had been added as a glue A record entry. After that, I made the client send the same DNS query using "dig @68.180.131.16 www.google.co.uk" again to the Internet. The results showed that the RYU controller passed the DNS query with "add flow table 68.180.131.16" message since the destination IP address "68.180.131.16" had been added in the NS_DB.

Moreover, I performed the additional feature evaluation in the following Fig 4.10 as a tested environment again. The objective showed that my proposed system can block a direct outbound DNS query which was sent to a C&C server in a real complicated network environment. I constructed a virtual environment as well as a real network on host OSs, generated a direct outbound DNS query which was sent to C2server and checked whether the direct outbound DNS query was blocked. Firstly, the destination IP address of C2Server was not registered in NS_DB. In addition, I constructed a HTTP server including perl program "query.pl" which sends a direct outbound DNS query to a C2server on the Web. This is one process to be a bot-infected PC in a real environment. In this condition, I run the controller program. Client01 was able to download query.pl by "wget http://web01.exampl.com/query.pl" from Web. When client01 run "query.pl", I checked blocking the DNS query with a TXT record sent to C2Server by the tcpdump command in C2Server. This result showed that the system worked well as I expected. Note that I maintain the NS record and glue A record entries for all domains including the root domain and the top level domains, which prevents unnecessary blocks of the DNS queries.

4.5.3 Performance evaluation

After completing the feature evaluation of NS_DB, I also evaluated the performance of the prototype using the experimental network shown in Fig. 4.10 in order to check the overhead of OpenFlow architecture in the proposed system. Note that 8.8.8.8 is registered in the NS_DB and the query has to be passed and passes the DNS query packet. The Client01 can send a DNS query to the DNS or a public DNS server on the Internet. I measured the latency of the DNS name resolution in various patterns and the detailed results are described in Tabs. 4.5 and 4.6.

First, I measured the latency of a single DNS query by using "dig" command from the Client01. As described in Tab. 4.5, when the Client01 used the internal DNS full resolver (172.16.101.20) the average latency in the conventional method (without using the prototype) for ten times was 515.5ms while that in the prototype was 530.7ms. By calculating the difference between the two types of latencies I confirmed the latency might be caused by the proposed system which was 15.2ms (530.7 515.5). In this case, I restarted the DNS before sending each DNS query from the Client01 which is indicating that the results were not cached in the DNS. When I used the cache in the DNS, I can see that all the latencies are "0" since all the DNS responses were sent back from the DNS. On the other hand, I also measured the latency with the same scenario using public DNS servers (8.8.8.8 and 1.1.1.1) and I found that, in both public DNS servers, the latency of the prototype was only a little more than that of a conventional method. I consider the reason was that when I used the DNS all the flow entries were added in the OpenFlow Switch by checking the OpenFlow Controller with "packet_in" function thus the latency was much higher. Moreover, as I can see that the standard deviation was also high in all evaluation cases and I consider the reason was that the network conditions were changing dynamically on the Internet thus the latency of each DNS query was also changed.

Next, I conducted performance evaluations using a DNS performance measurement tool named dnsperf 2.1.0 [64] which was installed on the Client01. In addition, the parameters of dnsperf "-s" identified target DNS IP address. I used one thousand domain names obtained from the QuantCast Measure [65] which provides a list of web sites. In this case, I only used the DNS and measured the query speed in [query/s] which indicates the number of queries sent per second when there was no packet loss which means all the

Table 4.7: List of public DNS servers

8.8.8.8, 8.8.4.4 (dns.google)
1.1.1.1, 1.0.0,1 (Cloudflare)
208.67.222.123 (opendns)
9.9.9.9 (quad9.net)
64.6.64.6, 64.6.65.6 (nstld.net)
216.146.35.35, 216.146.36.36 (dynect.net)
77.88.8.1, 77.88.8.2, 77.88.8.3, 77.88.8.88 (yandex.ru)
185.228.168.10, 185.228.168.11, 185.228.168.168, 185.228.168.169 (cleanbrowsing.org)
180.76.76.76 (baidu.com)
114.114.114.114 (114dns.com)
8.20.247.20 (dnsbycomodo.com)
161.97.219.84 (sourpuss.net)
104.168.144.17 (hostwindsdns.com)

one thousand domain names were processed successfully with DNS name resolution. As I can see from the Tab. 4.6, when I did not use the cache of the DNS, the average query speed of the proposed method was 40.14 [query/s] which was much less than that of the conventional method with 80.13 [query/s]. Here, when I add the limitation of maximum query speed (query/s) with 108 in the conventional method, I found that all the domain names were processed successfully while the value was 42 in the proposed method. This means the latency in the proposed method was much higher than that in the conventional method which shows the same results as in Tab. 4.5. On the other hand, when I allow the cache function of the DNS, in this case I did not indicate the limitation of max query speed, I can see that the query speed of the proposed method was about a half of that of the conventional method. I consider the reason was that the overhead of the OpenFlow architecture in the proposed method caused the low performance. However, I also think that it is possible to improve the performance of the proposed method by tuning up the OpenFlow Controller program as well as the matching method of the "packet_in" function and I plan to address this problem in my future work. Finally, as I can see from Tab. 4.6 when I limited the max query speed, the standard deviation was small while when I did not set the limitation the standard deviation was large. I consider the reason was that when there was no limitation on query speed the network condition can be easily changed so that the query speed can also be easily affected.

Table 4.8: False positive rate of the proposed system

	The number of destination IP addresses in pcap files (excluding registered glue A in NS_DB)	The number of destination IP addresses in pcap files	Ratio (%)
1st pcap file	1573	10773	14.60
2nd pcap file	1441	9599	15.01
3rd pcap file	1676	9694	17.28
4th pcap file	1647	9455	17.41
5th pcap file	1255	7095	17.68
6th pcap file	1395	9437	14.78
7th pcap file	1445	10170	14.20
8th pcap file	1460	10933	13.35
9th pcap file	1567	11239	13.94
10th pcap file	1544	10896	14.17
Avarage	N/A	N/A	15.25
Total	2479	26894	9.22

So far, my proposed method is basically an off-line detection method. To construct an online and pseudo real-time detection method, I can adjust the interval to create a single pcap file. For example, in my campus network, a two hours interval is corresponding to approximately 200MB of pcap file size. If a two hours interval does not satisfy a requirement from users, I can choose a smaller interval time, e.g., two minutes. Another important thing is the processing speed. Let's consider the following scenario with a two hours interval. In the first interval, the system records the first pcap file. In the second interval, the system updates NS_DB based on the first pcap file as well as it records the second pcap file. If the NS_DB update processing requires more than two hours, the system will not work as an online method. According to our experiments, the processing time becomes less than two hours, which is fast enough to construct an online system.

4.6 Discussion

In this thesis, to identify the abnormal DNS traffic used in botnet communication, I have proposed a detection system based on the observation; in a typical anomalous DNS traffic, a client computer sends a DNS query to the Internet directly without using the DNS full resolver. To identify such anomalous DNS traffic, I maintain a database of NS record

history. Based on this concept, I have designed, implemented a system and evaluated the feature and performance. One important issue in the proposed system is the database learning time (e.g. for one month), because the proposed system can decrease false positive/negative by storing many NS records and glue A records. Database entries will be updated dynamically, and if the database does not contain required NS record history entries, e.g., at the beginning phase of the system, then the false positive/negative may occur. To deal with this matter, I have to introduce the database learning phase.

1. Just after the system started, no DNS queries will be immediately blocked.

2. After the learning phase finished, anomalous DNS queries will be blocked.

I evaluated a metric in terms of the detection accuracy of my proposed. The evaluation metric was the false positive rate (FPR) in terms of malicious DNS queries, which is presented in Table 4.8. More specifically, I use the first ten pcap files in log data, which were obtained in my previous study [66]. First, I manually confirmed that the pcap files do not contain any malicious traffic. And, a pcap file of DNS traffic for about two hours approximately has a size of 200MB. In my proposed method, the destination IP address of DNS query packets will be registered in NS_DB if the query was legitimate. So, the destination IP address not registered in NS_DB was a false positive. In Table 4.8, I show the FPR of ten individual pcap files as well as the total FPR which was calculated using concatenated ten pcap files. As a result, the ratio was 9.22% (=2479/26894) in the total. 2479 IP addresses may include public DNS, and other legitimate use cases including the software update protocol. Note that the famous public DNS servers described in Table 4.7 were registered in NS_DB for decreasing the FPR, however other legitimate use cases may exist. Another source of the false positive is the division of multiple pcap files, i.e., after a change in saved pcap files, glue A record happens, so that the NS record has not been registered in NS DB. This is the reason why the total FPR is much larger than the individual FPRs. I can decrease this type of false positive to shorten the interval of pcap file saving.

Another way to decrease the false positive/negative rates is to introduce a monitoring system; the network administrator will cooperate with the automatic detection system.

The design and performance evaluation of the learning phase and the monitoring system will be future work.

4.7 Conclusion

According to many security reports and researches, several types of DNS-based botnet use direct outbound queries and responses (Q&R), which are different processes form the normal DNS name resolutions. Based on this observation, in this thesis, I proposed a detection and blocking system of DNS-based botnet communication using a NS record database. In order to differentiate the normal and abnormal DNS queries, I created a database which stores NS records and the corresponding glue A records. In the normal use cases, NS records and the corresponding glue A records will be obtained prior to sending DNS queries to them, whereas some types of DNS-based botnet communication will not confirm the principle necessarily. Therefore, by using my proposed system, the destination IP address of a DNS query in a normal DNS name resolution will be stored in the database and the packet will be passed through as is while the destination IP address of malicious DNS traffic will not be included in the database so that the traffic will be blocked.

Based on my proposed system, I also implemented a prototype system using SDN technologies and performed feature evaluations and a preliminary performance evaluation. The prototype system consists of a NS record history database (NS_DB), a replica of NS_DB, an Open vSwitch and a Ryu controller. I used MariaDB for the database system and customized the Ryu controller using Python programing language. Based on the feature evaluation results using the prototype system on a local network environment, I confirmed that my proposed system worked as I designed and it was expectable to detect and block some types of DNS-based botnet communication. Moreover, based on the preliminary performance results I consider that although the results did not show a good performance it is possible to improve it by tuning the Ryu program, and finally, the proposed system can also be deployed in a real network environment.

For the future work, I obtain a NS record and need to analyze the corresponding glue AAAA record. Thus, I plan to set IPv6 in an OS or a network environment. I plan to

tune up the Ryu controller program in order to improve the performance of the proposed system and feature evaluation as well as performance test on a real network environment. Furthermore, I also plan to expand the proposed system to apply for other types of DNS-based botnet communication as well as malicious DNS traffic.

Chapter 5

Conclusion

The objective of this study was to create the system to detect and block DNS-based botnet communication. To achieve the goal, two steps was performed. In first step, real DNS traffic was analyzed three types botnet communication; via-resolver DNS query, indirect outbound DNS query, direct outbound DNS query. Based on this analysis, consideration of detection and blocking DNS-based botnet communication was performed. In second step, the system to detect and block DNS-based botnet communication was implemented and evaluated.

In particular, I analyzed three types of DNS traffic: through via-resolver, indirect outbound and direct outbound communication in first step. I investigated the DNS TXT records obtained from the DNS full resolvers of my campus network for via-resolver type botnet communication and categorized its usages. The result indicated that the DNS TXT queries about 30.3%, which were categorized “Unconfirmed”, were identified as “highly suspicious” and it had about 43.9% common detection rate with a conventional security check site “Virustotal.com.”. For the other two types, I investigated all the DNS traffic, excluding the via-resolver DNS traffic, obtained at rhe gateway of the campus network. As the result of my analysis, I found that about 19% and 8% of destination IP addresses were identified as “highly suspicious” in indirect and direct outbound DNS TXT queries, respectively. I achieved that suspicious DNS traffic was identified by my analysis. Based on the analysis, the possibility of a comprehensive botnet detection system was considered and proposed a design for such a system. I showed that this analysis was new method in non-exiesting research.

Next, based on above proposed method, the system of detection and blocking DNS-based botnet communication was implemented in second step. The feature of the system was evaluated as expected. Furthermore, the performance evaluation was shown that the latency might be caused by the proposed system which was 15.2ms (530.7 - 515.5). Moreover, false positive ratio was 9.22% in the total. This was why this type of false positive was shorten the interval of pcap file saving. This system was shown the effectiveness. I achieved that my proposed system provided effectively detection and blocking DNS-based botnet communication.

In summary, the thesis studied the system for detecting and blocking DNS-based botnet communication by analyzing the achieved NS records history. The contribution of this study is to confirm new analytical method in DNS-based botnet communication and to create detection and blocking system in DNS-based botnet communication by my analysis.

In future work, network administrators and computer emergency response team need to provide secure network excluding various security attacks more than now. My contribution is simply detection and blocking DNS-based botnet communication as soon as possible in one of methods to keep secure network. The new DNS-based botnet communication may be appeared by evolution. Public DNS server or DNS full resolver in a organization also may be hijacked temporarily. Then, network administrators and computer emergency response team may judge legitimate DNS traffic. I consider that DNS server should aim to be secure and reliable more than now.

Bibliography

[1] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, and K. Beznosov, "Know your enemy: the risk of unauthorized access in smartphones by insiders," *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services,* pp. 271–280, Aug. 2013. DOI: 10.1145/2493190.2493223

[2] J. Mirkovic, and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review,* April. 2004. DOI: 10.1145/997150.997156

[3] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, and K. Beznosov, "Spam Mail Reduces Economic Effects," *Second International Conference on the Digital Society,* Sainte Luce, Martinique, pp. 20–24, Feb. 2008. DOI: 10.1109/ICDS.2008.15

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *IEEE Computer,* vol. 50, pp. 80–84, July. 2017. DOI: 10.1109/MC.2017.201

[5] J. Liu, Y. Xiao, and K. Ghaboosi, "Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures" *EURASIP Journal on Wireless Communications and Networking* vol. 2009, 11 pages, Sept. 2009. DOI: 10.1155/2009/692654

[6] Trend Micro, "Security Intelligence Blog" `https://www.bleepingcomputer.com/news/security/smominru-botnet-infected-over-500-000-windows-machines/`, Aug 2019.

[7] Trend Micro, "Security Intelligence Blog" `https://blog.trendmicro.com/trendlabs-security-intelligence/virobot-ransomware-with-botnet-capability-breaks-through/`, Sept 2018.

[8] The spamhaus project, "Botnet Threat Report 2019" `https://www.deteque.com/app/uploads/2019/02/Spamhaus-Botnet-Threat-Report-2019.pdf`, Feb. 2019.

[9] G. Ollmann, and Damballa inc., "Botnet communication topologies," online http://www.technicalinfo.net/papers/PDF/WP_Botnet_Communications_Primer_(2009-06-04).pdf, 2009.

[10] A. K. Sood, and S. Zeadally, "A Taxonomy of Domain-Generation Algorithms," in *IEEE Security & Privacy,* vol. 14, pp. 46–53, July-Aug. 2016. DOI: 10.1109/MSP.2016.76

[11] OpenDNS inc., "The role of DNS in botnet command and control," online http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitepaper-DNSRoleInBotnets.pdf, 2012.

[12] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in *7th IEEE International Conference on Computer and Information Technology (CIT 2007),* Aizu-Wakamatsu, Fukushima, Japan, pp. 715–720, Oct. 2007. DOI: 10.1109/CIT.2007.90

[13] J. Oikarinen, and D. Reed, "Internet relay chat protocol," IETF RFC1459, May 1993.

[14] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szyd-lowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in Proc. ACM Conference on Computer and Communications Security (CCS ' 09), pp. 635-647, Nov 2009.

[15] J.B. Grizzard, V. Sharma, C. Nunnery, B.B. Kang, and D. Dagon, "Peer-to-Peer Botnets: Overview and Case Study," in Proc. USENEX, Workshop on Hot Topics in Understanding Botnets (HotBots'07), 8 pages, April 2007.

[16] A.M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, "Detection of malicious payload distribution channels in DNS," in *Proc. IEEE Int'l Conference on Communications (ICC2014),* Sydney, Australia, pp. 853–858, June 2014. DOI: 10.1109/ICC.2014.6883426

[17] K. Xu, P. Butler, S. Saha, and D. Yao, "DNS for massive-scale command and control," *IEEE Trans. Dependable and Secure Computing,* vol. 10, no. 3, pp. 143–153, May/June 2013. DOI: 10.1109/TDSC.2013.10

[18] C.J. Dietrich, C. Rossow, F.C. Freiling, H. Bos, M. Steen, and N. Pohlmann, "On botnets that use DNS for command and control," in *Proc. IEEE European Conference on Computer Network Defence (EC2ND'11),* Gothenburg, Sweden, pp. 9–16, Sept. 2011. DOI: 10.1109/EC2ND.2011.16

[19] J. Damas, and P. Vixie, "Extension Mechanisms for DNS (EDNS0)," *IETF RFC6891,* April. 2013.

[20] S. Kitterman, "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1," *IETF RFC7208,* April 2014.

[21] D. Crocker, T. Hansen, M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures," *IETF RFC6376,* September 2011.

[22] C. Mullaney, "Morto worm sets a (DNS) record," http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record, Aug. 2011.

[23] Google, "Introduction to Google public DNS," https://developers.google.com/speed/public-dns/docs/intro, Accessed on May 30 2016.

[24] D. Kaminsky, "The black ops of DNS," Presented in Black Hat USA 2004, Las Vegas, NV, July 2004. http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-kaminsky/bh-us-04- kaminsky.ppt

[25] AnalogBit, "tcp-over-dns," http://analogbit.com/software/tcp-over-dns, last accessed on July 6 2016.

[26] S. Bromberger, "DNS as a covert channel within protected networks," *White paper of Department of Energy*, http://energy.gov/oe/downloads/dns-covert-channel-within-protected- networks, Jan. 2011.

[27] J. Riden, "How fast-flux service networks work," http://www.honeynet.org/node/132, Accessed on May 30 2016.

[28] M. Singh, and S. Kaur, "Issues and Challenges in DNS based Botnet Detection: A Survey," *Computers & Security,*, Vol. 12, pp. 28–52, Elsevier (online), DOI: 10.1016/j.cose.2019.05.019 (2019).

[29] G. Farnham, "Detecting DNS tunneling," *White paper of SANS institute,* https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152, 32 pages, March 2013.

[30] J. Ahmed, H. Gharakheili, Q. Raza, C. Russell and V. Sivaraman, "Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks," *Proc. IFIP/IEEE Symposium on Integrated Network and Service Management 2019* (IM2019), pp. 649–653 (online) http://dl.ifip.org/db/conf/im/im2019short/188679.pdf (2019).

[31] S. Khattak, N.R. Ramay, K.R. Khan, A.A. Syed, and S.A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Commun. Surveys & Tutorials,* vol. 12, no. 2, pp. 898–924, Oct. 2013. DOI: 10.1109/SURV.2013.091213.00134

[32] H. Binsalleeh, "Botnets: Analysis, detection, and mitigation," in *Network Security Technologies: Design and Applications,* ed. A. Amine, O.A. Mohamed, and B. Benatallah, pp. 204–223, IGI Global, Hershey, PA, Nov. 2013. DOI: 10.4018/978-1-4666-4789-3.ch012

[33] S. Soltani, S.A.H. Seno, M. Nezhadkamali, and R. Budiarto, "A survey on real world botnets and detection mechanisms," *Int'l Journal of Information and Network Security,* vol. 3, no. 2, pp. 116–127, Apr. 2014.

[34] McAfee, "McAfee labs threats report," `http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-mar-2016.pdf`, March 2016.

[35] P. Mockapetris, "Domain names: Concepts and facilities," *IETF RFC1034*, Nov. 1987.

[36] P. Mockapetris, "Domain names: Implementation and specification," *IETF RFC1035*, Nov. 1987.

[37] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proc. IEEE Int'l Conference on Emerging Security Information, Systems and Technologies,* pp. 268–273, Glyfada, Greek, June 2009. DOI: 10.1109/SECURWARE.2009.48

[38] N.M. Hands, B. Yang, and R.A. Hansen, "A study on botnets utilizing DNS," in *Proc. ACM Conference on Research in Information Technology (RIIT'15),* pp. 23–28, Chicago, IL, USA, Sept.-Oct. 2015. DOI: 10.1145/2808062.2808070

[39] M. Anagnostopoulos, G. Kambourakis, and S. Gritzalis, "New facets of mobile botnet: Architecture and evaluation," *Int'l Journal of Information Security,* 19 pages, Dec. 2015. DOI: 10.1007/s10207-015-0310-0

[40] Sophos Ltd., "Overview of the Sophos live protection architecture in SESC 9.5+,", `https://www.sophos.com/en-us/support/knowledgebase/111334.aspx`, Accessed on May 30 2016.

[41] Cisco, "Immunet 3.0," `http://www.immunet.com/`, Accessed on May 30 2016.

[42] AVG, "AVG internet security 2015," `http://www.avg.com/us-en/internet-security`, Accessed on May 30 2016.

[43] SpamCop, "How can I check if an IP is on the list?" https://www.spamcop.net/fom-serve/cache/351.html, Accessed on May 30 2016.

[44] Spamhaus project, "Spamhaus," https://www.spamhaus.org, Accessed on May 30 2016.

[45] J. Hildebrand, P. Saint-Andre, and L. Stout, "XEP-0156: Discovering alternative XMPP connection methods," http://xmpp.org/extensions/xep-0156.html, Accessed on May 30 2016.

[46] Google, "Virustotal," https://www.virustotal.com/en/, Accessed on May 30 2016.

[47] H. Ichise, Y. Jin, and K. Iida, "Analysis of via-resolver DNS TXT queries and detection possibility of botnet communications," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM2015)*, pp. 216–221, Aug. 2015. DOI: 10.1109/PACRIM.2015.7334837

[48] H. Ichise, Y. Jin, and K. Iida, "Analysis of via-resolver DNS TXT queries and detection possibility of botnet communications," *IEICE Commun. Express,* vol. 5, no. 3, pp. 74–78, March 2016. DOI: 10.1587/comex.2015XBL0186

[49] W. Hardaker, "Child-to-Parent Synchronization in DNS," *IETF RFC7477,* March. 2015.

[50] Symantec, Inc., "What is a zero-day vulnerability?," http://www.pctools.com/security-news/zero-day-vulnerability/, Accessed on May 30 2016.

[51] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," *IETF RFC7858,* May. 2016.

[52] McAfee Labs, "Threats Report," https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2018.pdf, Accessed on December 10 2018.

[53] 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. DOI: 10.1587/transcom.2017ITP0009

[54] Google, "Introduction to Google Public DNS," https://developers.google.com/speed/public-dns/docs/intro, Accessed on December 10 2018.

[55] Cloudflare, "Introduction to Cloudflare Public DNS," https://www.cloudflare.com/learning/dns/what-is-1.1.1.1/, Accessed on December 10 2018.

[56] Open Networking Foundation, "SDN Architecture," https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf, Accessed on December 10 2018.

[57] S. Denazis, et al, "Software-Defined Networking (SDN): Layers and Architecture Terminology," *IRTF RFC7426,* January. 2015.

[58] Project Floodlight, "Floodlight," http://www.projectfloodlight.org, Accessed on December 10 2018.

[59] Python, "Python," https://www.python.org, Accessed on December 10 2018.

[60] NTT, "Ryu: Getting Started," https://osrg.github.io/ryu-book/en/html/, Accessed on December 10 2018.

[61] MariaDB, "MariaDB," https://mariadb.org, Accessed on December 10 2018.

[62] Open vSwitch, "Open vSwitch," https://www.openvswitch.org, Accessed on December 10 2018.

[63] Galera Cluster, "Galera Cluster," http://galeracluster.com, Accessed on December 10 2018.

[64] Akamai, "Measurement Tools," https://www.akamai.com/us/en/products/network-operator/measurement-tools.jsp, Accessed on December 10 2018.

[65] QuantCast, "Measure Audiences (online)" https://www.quantcast.com, Accessed on December 10 2018.

[66] Y. Jin, H. Ichise, and K. Iida, "Design of detecting botnet communication by monitoring direct outbound DNS queries," in *Proc. IEEE Int'l Confrerence on Cyber Security and Cloud Computing (CSCloud2015),* New York, NY, pp. 37–41, Nov. 2015. DOI: 10.1109/CSCloud.2015.53

Achievements

Publications forming part of the the thesis

A. Journal papers

[A-1] H. Ichise, Y. Jin, K. Iida, and Y. Takai "NS record History Based Abnormal DNS traffic Detection Considering Adaptive Botnet Communication Blocking," To appear in IPSJ Journal of Information Processing, 11 pages, Feb. 2020.

[A-2] 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.
DOI:10.1587/transcom.2017ITP0009

[A-3] H. Ichise, Y. Jin, and K. Iida, "Analysis of Via-resolver DNS TXT Queries and Detection Possibility of Botnet Communications," IEICE Commun. Express, vol. 5, no. 3, pp. 74-78, Mar. 2016.
DOI:10.1587/comex.2015XBL0186

B. International conferences (Peer-review)

[B-1] H. Ichise, Y. Jin, K. Iida, and Y. Takai, "Detection and Blocking of Anomaly DNS Traffic by Analyzing Achieved NS Record History," in *P*roc. Asia-Pasific Signal and Information Processing Association, Annual Summit and Conference 2018 (APSIPA-ASC2018), pp. 1586–1590 (online) `http://www.apsipa.org/proceedings/2018/pdfs/0001586.pdf` (2018).

[B-2] Y. Jin, H. Ichise, and K. Iida, "Design of detecting botnet communication by monitoring direct outbound DNS queries," in *Proc. IEEE Int'l Confrerence on Cyber Security and Cloud Computing (CSCloud2015),* New York, NY, pp. 37–41, Nov. 2015. DOI: 10.1109/CSCloud.2015.53

[B-3] H. Ichise, Y. Jin, and K. Iida, "Analysis of via-resolver DNS TXT queries and detection possibility of botnet communications," in *Proc. IEEE Pacific Rim Conference*

on Communications, Computers and Signal Processing (PACRIM2015), pp. 216–221, Aug. 2015. DOI: 10.1109/PACRIM.2015.7334837

[B-4] H. Ichise, Y. Jin, and K. Iida, "Detection Method of DNS-based Botnet Communication using Obtained NS Record History," in *Proc. IEEE Int'l Computers, Software and Applications Conference (COMPSAC2015) Workshops, Fast Abstract Track*, pp. 676-677, July. 2015. DOI: 10.1109/COMPSAC.2015.132