一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2017-31 (2017-11)

Design and Implementation of NS Record History Database for Detecting DNS-based Botnet Communication

Hikaru Ichise†, Yong Jin††, Katsuyoshi Iida†††

† Technical Department, Tokyo Institute of Technology, Japan and
Graduate School of Information Science and Technology, Hokkaido University, Japan
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8550 Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology, Japan
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8550 Japan
††† Information Initiative Center, Hokkaido University, Japan
Kita 11, Nishi 5, Kita-ku, Sapporo, 060-0811, Japan

Abstract DNS (Domain Name System) based domain name resolution service is one of the most fundamental Internet services for the Internet users and application service providers. In normal DNS based domain name resolution, the corresponding NS records are required in prior to sending DNS query to the corresponding authoritative DNS servers. However, in recent years, DNS based botnet communication has been observed in which botnet related network traffic is transferred via DNS packets. In particular, sending DNS queries to C&C servers using IP address directly without obtaining the corresponding NS records is present in some malware. In this paper, we focus on this type of botnet communication and design a NS record history database for detecting DNS-based botnet communication. We implemented a prototype system and evaluated the feature of NS records history creation as well as the checking function. Based on the evaluation results we confirmed the proposed database worked as we designed and it is expected to detect the target botnet communication.

Key words Botnet communication, DNS, NS record history, database

1 Introduction

Recently, botnet based cyber attacks have become one of the biggest security issues for Internet users [1]. In addition to the conventional cyber attack technologies, recently we have observed that some types of botnet program used DNS (Domain Name System) protocol. DNS protocol is one of the most widely used communication protocols in the Internet, which is mainly used for domain name resolutions. Many features of DNS provide useful services to the Internet users and one of the fundamental and important features is translating domain name to IP address and vice versus. Besides this basic feature, DNS also has many other types of resource records such as TXT, SRV, DNSSD, etc [2] providing corresponding services. Unfortunately, we also observed some types of botnet program used DNS protocol for their communication and cyber attacks. Especially for the DNS TXT resource record type, it has free format in plain text with more than 4,000 Bytes of capacity and therefore is convenient for transferring information [2].

Several researches have reported that DNS TXT record type is used on botnet for some purposes [3]. In the prior stage of our research, we have analyzed a set of real DNS traffic obtained from our campus network to develop a method for detecting DNS based botnet communication. In our first round of analysis, we focused on the usages of DNS TXT record type and investigated the real DNS traffic captured from the DNS full resolvers of our campus network. Based on the analysis results, we managed to categorize the legitimate usages of DNS TXT record type so that it became expectable to detect botnet communication by checking the categories [4, 5].

In the next analysis, we found that some botnet programs used DNS full resolvers (A recursive DNS server for providing name resolution service for clients), while some did not for their botnet communication. We call the DNS queries using DNS full resolvers via-resolver DNS queries and those not using DNS full resolvers direct/indirect outbound DNS queries. We captured a set of real DNS traffic from border router of our campus network and analyzed the unique destination IP addresses of DNS TXT record in direct/indirect outbound DNS queries which were sent out without using DNS full resolvers as well. As a result, we confirmed that 19% of direct outbound DNS queries and 8% of indirect outbound DNS queries have been malicious communication by using third party web site. Based on the analysis results, we have designed an automatic detection system for DNS-based botnet communication by monitoring direct/indirect outbound DNS queries [6] [1]. Considering the traffic control features of SDN (Software Defined Network), in the design, we selected SDN architecture for traffic control [9]. One of the important factor of the system is NS records history since the direct/indirect

[1]The paper [7] is a unified version of the paper that includes both the first analysis [4, 5] and the second analysis [6].

outbound DNS queries before never having obtained authoritative DNS records are like to be suspicious DNS traffic. Thus the construction of effective database for NS records history is important to realize the system.

In this paper, we focus on the design and implementation of legitimate NS record history database for detecting DNS-based botnet communication. In Sect. 2, we will introduce three types of DNS-based botnet communication; via-resolver DNS query, direct outbound DNS query, indirect outbound DNS query. In Sect. 3, we will describe the design of database for storing NS record history in detail. In Sect. 4, we will explain the implementation of a prototype system. In Sect. 5, we will describe evaluation with creating a sample database using real DNS traffic by using the prototype system. Finally, we will conclude the paper in Sect. 6.

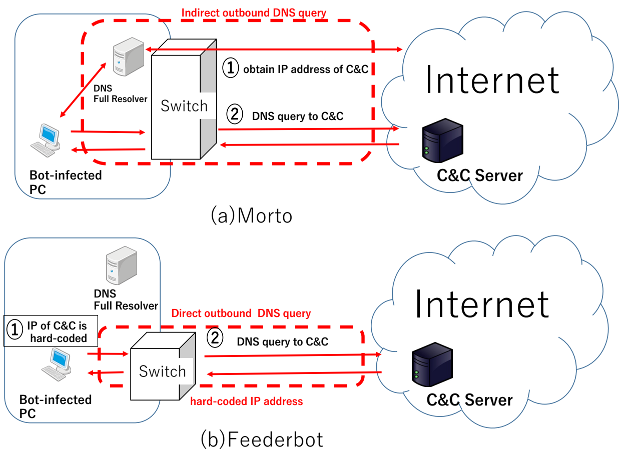

Figure 1: Direct & indirect outbound DNS queries

2 DNS-based botnet communication and related work

In this section, we 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. Then we introduce some related researches.

2.1 Three types of DNS-based botnet communication

In the network of an organization, we usually set up one or more DNS full resolvers for providing DNS name resolution service to the internal computers. The internal computers send DNS queries to the DNS full resolvers and the DNS full resolvers perform name resolution on behalf of those internal computers. As we mentioned before, once a computer is infected by some botnet program, the computer needs to contact with its C&C server which is called botnet communication.

We find out that there are three types of DNS-based botnet communication; using via-resolver DNS query, using indirect outbound DNS query and using direct outbound DNS query. The way of using via-resolver DNS query relies on DNS full resolvers completely. In [10], the authors have reported the usage of DNS TXT record type in botnet communication using via-resolver DNS query. Next, botnet communication using indirect outbound DNS query is famous for Morto [14]. Fig. 1(a) shows that the bot-infected PC obtains the IP address of its C&C server by DNS based name resolution via the DNS full resolver at first, then sends DNS queries to C&C server directly. On the other hand, Feederbot which is another type of botnet program, never uses DNS full resolvers, which called this as direct outbound DNS query [11]. Fig. 1(b) shows that the bot-infected PC obtains the hard-coded IP address of C&C server when the botnet program is istalled, then sends DNS queries to C&C server using the hard-coded IP address directly. These two types of DNS-based botnet communication that used in Morto and Feederbot, never obtain authoritative NS record before sending DNS query to the C&C servers. Therefore we consider that if we could collect the obtained authoritative NS records and check the destination IP addresses of all DNS queries in the network of an organization, it will be possible to detect these types of DNS-based botnet communication.

2.2 Related work

Since DNS-based botnet communication might be used widely, there were some researches in the literature. In [12], the authors introduced DNS tunneling technology in which the malicious contents were included in the labels of domain names. In order to detect such DNS tunneling, the authors discussed two methods, payload analysis and traffic analysis. In payload analysis, they analyzed for tunnel indicators from DNS payload in DNS query and response. In traffic analysis, they analyzed over time in DNS traffic.

In [11], the authors analyzed 14 milloin DNS TXT traffics and detected a new botnet, "Feederbot". The authors combined protocol-aware information theoretical features with behavioral communication features and applied them at different levels of network traffic abstraction. Moreover, they classified malware concerning DNS C&C usage based on network traffic.

However, none of the existing researches considered NS record history obtained from the received DNS responses. Thus, we tend to create automatic detection system for DNS-based botnet communication by constructing the legitimate NS record history database and monitoring all DNS queries.

3 Design of NS record history database

As described in the previous sections, in the recent DNS-based botnet communication, the bot-infected computer sends DNS queries to its C&C servers without obtaining corresponding NS records. Thus we proposed a detection method for DNS-based botnet communication using legitimate NS record history. In this section, we describe the detail of NS record history database and management procedures.

3.1 Overview

In DNS based name resolution, computers or DNS full resolvers send DNS query to the Internet and receive response. In general, corresponding NS records will be included in the responses which will received during the name resolution. Thus we need to capture all DNS query and response pairs from the network of an organization to obtain the NS record history. Specifically, we considered to create a temporary database for DNS queries named query_DB first, then capture all DNS responses corresponding to the DNS queries stored in query_DB. Finally, we pick up all NS record history and the IP addresses of corresponding name servers for creating the final database NS_DB for NS record history. In the following sections, we describe the temporary database query_DB first followed by the detail of database NS_DB.

3.2 Query database

First, we pick up all DNS queries from the captured tcpdump files and create a temporary database query_DB. We selected the following data field of each DNS query of the tcpdump file and input them to the database query_DB.

- querytime: The UNIX time in milisecond when the DNS query is sent.
- queryid: The message id included in DNS query.
- queryname: The FQDN and record type of DNS query. For example, when the query name in the DNS query is '251.13.112.131.in-addr.arpa' and the resource record type is 12, the FQDN means '251.13.112.131.in-addr.arpa' and the record type '12' means DNS PTR record. But if the record type is A which means IP address, this data field only includes FQDN.

3.3 NS record history database

After the query_DB is created, we filter out all the DNS responses from the same tcpdump file based on which the query_DB is created and pick up corresponding DNS response for each DNS query. Consequently, we pick all DNS query and response pairs from the tcpdump file. Then we pick up all NS record history from the DNS responses and store them in the NS_DB. In general, NS records can be obtained from the authority section of the DNS response and corresponding glue A recored [2]

[2]Glue A record indicates the IP address of the corresponding name server.

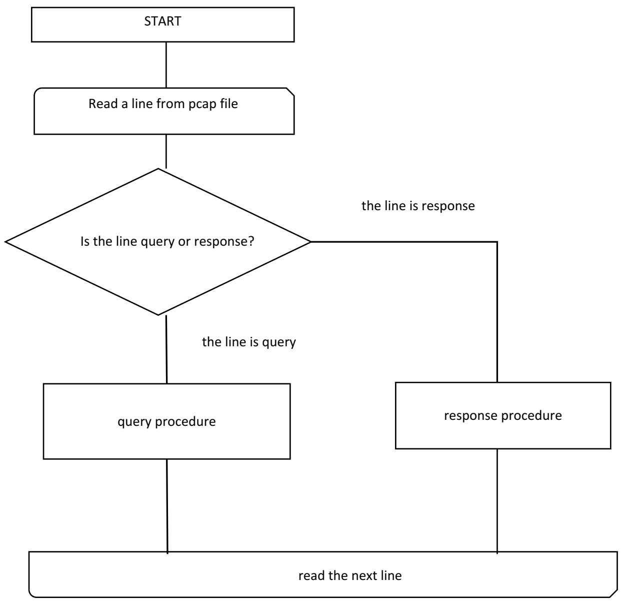

Figure 2: Flow chart of NS_DB operations

can be obtained from the additional section. However, in some cases, the additional section does not include corresponding glue A records though the authority section has some NS records. In such cases, we need to search the glue A record in the continuous DNS query and response pairs. The following data field are created in the NS_DB.

- res_time: The UNIX time in milisecond when the response is received.
- zname: The zone name for which a DNS server is authoritative.
- nsttl: The TTL (Time To Live) in second of the NS record.
- nsfqdn: The FQDN of a name server included in a ns record.
- glueAttl: The TTL in second of the glue A record.
- glueA: The IP address of the name server.

3.4 Detailed procedure of database management

Fig. 2 illustrates the detail flow chart of NS_DB operations. First of all, the tcpdump file will be read by the program which creates the database per line. Then if the line is a DNS query, the program performs procedures for DNS query otherwise performs the procedures for DNS response. In query procedure, a new entry will be added to the query_DB while in the response procedure a new entry will be added to NS_DB. Note that some DNS responses have NS record with glue A record in the additional section while some others only have NS records without glue A records in the additional section.

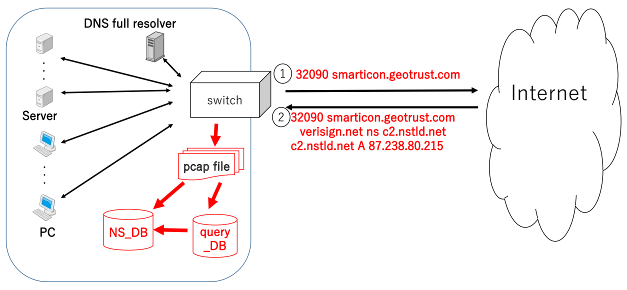
DNS full resolver
Server
PC
switch
pcap file
NS_DB
query _DB
1 32090 smarticon.geotrust.com
2 32090 smarticon.geotrust.com
verisign.net ns c2.nstld.net
c2.nstld.net A 87.238.80.215
Internet

Figure 3: Network topology for obtained data

In case of that the DNS response only has NS records, zname, nsttl and nsfqdn are registered to NS_DB. After that, if the DNS query for the IP address of nsfqdn has been sent then the received A record will be registered as the glue A record of the nsfqdn in the NS_DB.

4 Implementation

We implemented a prototype system using python program language and MariaDB database. In addition, we also used dpkt and pymsql python modules for processing tcpdump files and managing MariaDB.

Figure 3 shows the network topology for obtaining DNS traffic. The obtained tcpdump files include all DNS queries (the arrows with number 1 in Fig 3) and DNS responses (the arrow with number 2 in Fig 3) captured from our campus network. Firstly, the dpkt module decodes the tcpdump file. In this step, for example, the Queryid “32090” and queryname “smarticon.geotrust.com” from decoded tcpdump file are registered to the query_DB using the pymysql module. Zname “verisign.net”, Nsttl “900000”, nsfqdn “c2.nstld.net”, glueAttl “23570000”, glueA “192.26.92.31” from decoded pcap file to query_DB are registered to NS_DB of MariaDB using the pymysql.

Table 1 shows an entry of the NS_DB created by the prototype program. Note that if the DNS response only has NS record, the data fields of zname, nsttl, and nsfqdn of the NS record will be registered in the NS_DB but the those of glueAttl and glueA will be empty. After that, if the corresponding glue A record will be received within two seconds, the glueAttl and glueA will be registered in the NS_DB. We also need to register some exceptions. For example, we need to register IP addresses of the public DNS servers since those IP addresses belong to legitimate external DNS full resolvers. We also created an evaluation program using python in order to evaluate the prototype system and we describe the evaluation in the following section.

5 Evaluation

We used a tcpdump file with size of approximately 200MB to crate the NS_DB for the evaluation and 18,079 NS records were created in the NS_DB. Fig. 4 shows

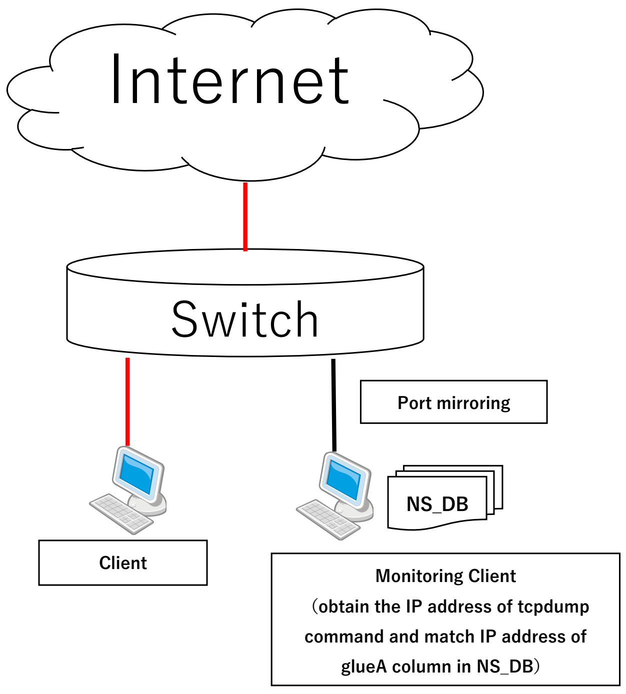
Internet
Switch
Port mirroring
NS_DB
Client
Monitoring Client
(obtain the IP address of tcpdump command and match IP address of glueA column in NS_DB)

Figure 4: Evaluation topology

the local network topology for the evaluation. We sent DNS query from the client to some specified destination IP address and monitored all the DNS traffic on the monitoring client. When the monitoring client obtained a DNS query, it checks the destination IP addresses of the DNS query in the database NS_DB. If the destination IP address exists in the glueA column of the NS_DB, the monitoring client prints out “<destination IP address>is registered glueA record” otherwise “<destination IP address>is unknown”. Consequently we can consider that the DNS queries with “<destination IP address>is unknown” can be botnet communication. The commands we used fro sending DNS queries and the results are shown in Table 2. As shown in the table, we used 3 FQDNs as samples; google.com, verisign.net and akamai.net. For the destination IP addresses, we used google public DNS resolvers 8.8.8.8 and 8.8.4.4 which were not registered in NS_DB and other three IP addresses which were registered in the NS_DB. According to the evaluation results, we can see that the DNS queries sent to those not registered in the NS_DB can be detected as we expected while to those registered in NS_DB also can be filtered out correctly.

6 Conclusion and Future work

In this paper, we designed and implemented a NS record history database for detecting DNS-baed botnet communication. We implemented a prototype system using python program language and MariaDB. We also created an evaluation program and evaluated the prototype system using real DNS traffic data. Based on the evaluation results, we confirmed the prototype system

Table 1: Columns of NS record history database

querytime	queryid	queryname	res_time	zname	nsttl	nsfqdn	glueAttl	glueA
1414782044594	32090	smarticon.geotrust.com	1414782044659	verisign.net	900000	c2.nstld.net	23570000	192.26.92.31

Table 2: Results of Feature evaluation

run the command in client	result of checking destination IP address by NS_DB
nslookup www.google.com 8.8.8.8	8.8.8.8 is unknown
nslookup www.google.com 8.8.4.4	8.8.4.4 is unknown
nslookup www.verisign.net 192.26.92.31	192.26.92.31 is registered glueA record
nslookup www.verisign.net 192.12.94.31	192.12.94.31 is registered glueA record
nslookup www.akamai.net 23.61.249.54	23.61.249.54 is registered glueA record
nslookup www.akamai.net 23.61.249.55	23.61.249.55 is registered glueA record

worked as we designed and it is expectable to detect DNS-based botnet communication. The future work includes the evaluation of the NS record history database in actual network environment.

References

[1] 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

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

[3] 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.

[4] 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

[5] 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

[6] 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

[7] H. Ichise, Y. Jin, and K. Iida, "Analysis of DNS TXT record usage and consideration of botnet communication detection," to appear in *IEICE Trans. Commun.*, vol. E101-B, no. 1, 10 pages, Jan. 2018. DOI: 10.1587/transcom.2017ITP0009

[8] H. Ichise, Y. Jin, and K. Iida, "Detection method of DNS-based botnet communication using obtained NS record history," *Proc. IEEE International Computers, Software & Applications Conference (COMPSAC2015) Workshops,* Fast Abstract Track, pp. 676-677, July. 2015. DOI: 10.1109/COMPSAC.2015.132

[9] S. Li, Y. Jin, and K. Iida, "Detection and control of DNS-based botnet communications by using SDN-Ryu solution," *IEICE Tech. Rep.,* vol. 115, no. 482, IA2015-93, pp. 73-78, March 2016.

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

[11] 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

[12] G. Farnham, "Detecting DNS tunneling," *White paper of SANS institute,* Mar. 2013.

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

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