Detection and Blocking of DGA-based Bot Infected Computers by Monitoring NXDOMAIN Responses

Yuki Iuchi
Graduate School of Information Science and Technology,
Hokkaido Univ.
Sapporo, Japan
yuki0311@frontier.hokudai.ac.jp

Yong Jin
Global Scientific Information and Computing Center,
Tokyo Inst. Tech.
Tokyo, Japan
yongj@gsic.titech.ac.jp

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

Katsuyoshi Iida and Yoshiaki Takai
Information Initiative Center,
Hokkaido Univ.
Sapporo, Japan
{iida,takai}@iic.hokudai.ac.jp

***Abstract*—Cyberattacks by botnets keep on increasing. In this research, we aim to detect and block Domain Generation Algorithm (DGA)-based bot-infected computers by focusing on the characteristics of domain name resolution for searching the Command & Control (C&C) servers. The attackers register only few of the DGA-based domain names for the C&C servers and make the bot-infected computers search them using DNS domain name resolution for the further instructions. This makes the DNS domain name resolution in C&C server searching process inevitably causing NXDOMAIN responses for queries about nonexistence domain names. In this paper, we designed and implemented a detection and blocking system against DGA-based bot-infected computers searching for the C&C servers by analyzing the DNS traffic resulted with NXDOMAIN responses. According to the feature evaluation results, we confirmed that the prototype system was effective for multiple types of DGA-based bots thus the approach could be applicable to detect and block the malicious DNS traffic from the bot-infected computers at the early stage.**

***Index Terms*—Bot, DNS, DGA, NXDOMAIN, and SDN.**

I. Introduction

The number of cyberattacks such as Distributed Denial of Service (DDoS) attacks and confidential data breaches keep on increasing nowadays. For example, in 2019, Emotet had been widely spread and many security organizations such as JPCERT/CC had issued alerts [1]. Many of these cyberattacks are considered to be conducted by botnets. When a computer is infected by a bot program, it first searches for its Command and Control (C&C) servers for the further instructions. Then many bot-infected computers form a botnet and carry out cyberattacks based on the instructions of the C&C servers [2]. Apparently, the detection and blocking of the communication between the bot-infected computers and the C&C servers at the early stage of the infection are important to protect it from the further cyberattacks.

The life cycle of a botnet can be divided into five phases [3]. In the first two phases, the bot programs invade computers through various routes and prepare the environment to complete the bot infection. Then the bot-infected computers move to the third phase, in which, a communication channel with the C&C servers (a botnet channel) will be established. In the last two phases, malicious actions will be conducted and the bot programs will be updated upon receiving instructions from the C&C servers. So far it has been clarified that Internet communication protocols like Internet Relay Chat (IRC), Peer-to-Peer (P2P) and Domain Name System (DNS) for botnet channels and many researches in the literature and, have been conducted [4]–[6] for the detection and blocking.

However, the establishment of a botnet channel requires identifying the C&C servers first. This means that it is possible to detect and block a bot-infected computer at this phase before the cyberattacks carried out based on the instructions of the C&C servers. DNS, a protocol for domain name resolution such as mapping domain names and IP addresses, has been generally used for identifying the C&C servers since distinguishing malicious DNS traffic from normal one is difficult and the network administrators cannot simply block all DNS traffic. So far the blacklist (including domain names and IP addresses) based approaches have been widely adopted for the detection and blocking of the communication between a bot-infected computer and the C&C servers [2]. However, these approaches have become ineffective since the attackers started using a technology named Domain Generation Algorithms (DGA) that generates many random domain names for the C&C servers. Consequently, only few of the domain names will be registered in the authoritative DNS servers for the C&C servers thus, the identification is significantly difficult.

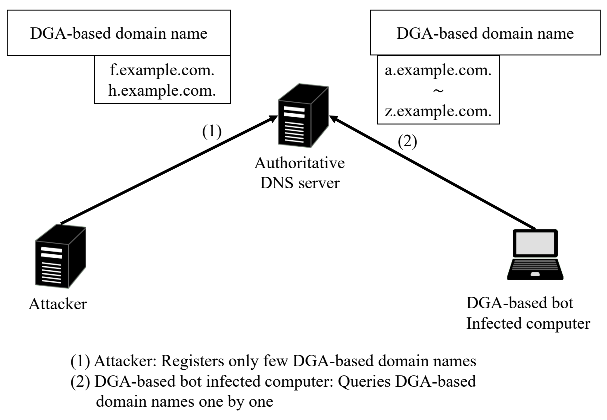
DGA-based domain name
f.example.com.
h.example.com.
DGA-based domain name
a.example.com.
~
z.example.com.
(1)
(2)
Authoritative DNS server
Attacker
DGA-based bot Infected computer
(1) Attacker: Registers only few DGA-based domain names
(2) DGA-based bot infected computer: Queries DGA-based domain names one by one

Fig. 1. A mechanism of identifying C&C servers

Figure 1 shows the basic mechanism of a DGA-based bot-infected computer identifying the C&C servers. First, an attacker registers only few domain names generated by a DGA (step (1)) and creates a bot program involving the same DGA for spreading. When a computer is infected by the bot program, the computer generates domain names using the DGA and queries them one by one until it hits active one registered in the authoritative DNS server (step (2)). Since different type of bot program uses different type of DGA thus distinguishing the DNS traffic generated by the bot-infected computers from the normal DNS traffic is difficult.

Fortunately, the DNS domain name resolution for searching C&C servers on bot-infected computers has a unique characteristic that the queries cause many NXDOMAIN responses which means that the queried domain names have not been registered in the authoritative DNS server. With focusing on this characteristic, the purpose of this research is to detect and block the DNS traffic from the DGA-based bot-infected computers for searching the C&C servers that can effectively protect it from the cyberattacks by the botnet.

The rest of the paper is organized as follows. Section II introduces DGA and related work followed by the proposed system and the prototype implementation in Sect. III. Section IV describes the evaluations and results followed by the summary of the paper and some future work in Sect. V.

II. DGA and related work

A. DGA

DGA is an algorithm used for generating many random domain names automatically. Figure 2 shows the flow chart of the domain name resolution process of identifying the C&C servers on a DGA-based bot-infected computer.

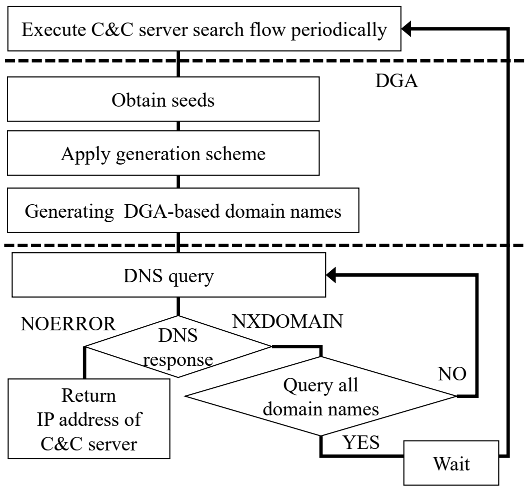
Execute C&C server search flow periodically
DGA
Obtain seeds
Apply generation scheme
Generating DGA-based domain names
DNS query
NOERROR
DNS response
NXDOMAIN
Return IP address of C&C server
Query all domain names
NO
YES
Wait

Fig. 2. The flow of DNS domain name resolution for searching C&C servers

As shown in Fig. 2, a DGA-based bot-infected computer periodically repeats a certain cycle of DNS domain name resolution until it obtains the IP addresses of the C&C servers. This is a cycle to identify the C&C servers by querying the domain names generated based on the same DGA as used by the attacker. If the bot-infected computer succeeds in identifying the C&C servers, which means the bot-infected computer obtains the IP addresses of the C&C servers, it breaks out from the domain name resolution cycle and establishes a botnet channel with the C&C servers for achieving the further instructions. On the other hand, if the bot-infected computer fails to obtain the IP addresses of the C&C servers, which means that the domain name resolution ends with NXDOMAIN response, it enters a waiting state and waits for the next cycle to be executed. The waiting period depends on the type of DGA (in terms of seed changes) and the number of domain names generated in one cycle, etc. DGA has been used in many types of botnets and one of the most famous botnets has been spread in recent years is Mirai [7] which was first observed in 2016. Mirai has many variants because its source code is publicly available.

As mentioned above, there are various variants of botnets. Similarly, DGA also has various ways of generating domain names, which are classified into two categories based on the seed source and generation scheme [8]. The seed is what the DGA uses to generate random domain names while the seed source is used for determining the seed. We focus on the time-dependent deterministic (TDD) type DGAs since this type is the most popular one [2], [8]. A typical example of the seed used by the TDD type DGAs is the current time. The current time is time-dependent and of course the attacker can easily predict it. Another time-dependent type of DGA, time-dependent non-deterministic (TDN), uses online information changes dynamically such as stock market charts and trending keywords on twitter whose future values are hard to be predicted, to generate domain names. In this paper, we only focus on the TDD type of DGAs.

B. Related work

Although there have been various researches conducted on DGA, there are mainly two research areas about its analysis: domain name centric [9] and the bot behavior centric [10], [11]. In [9], domain names were analyzed using the Jaccard index, Edit distance, and Kullback-Leibler divergence. The results of the statistical analysis are used to discriminate the communication related to DGA-based domain names. In [10], the authors clustered similar strings from the queried domain names that resulted with NXDOMAIN responses in a designated network and filtered DGA-based domain names in order to detect bot-infected computers. In [11], the authors also used linguistic features of the domain names and 27 features of the DNS protocol to discriminate DGA-based domain names and conducted machine learning-based bot detections.

However, the above researches require a certain amount of learning time and training data. As a result, if the behavior of bots is updated during training phase the detection accuracy will be reduced thus the system should be trained again with new data to respond to the latest DGAs. Also, there is no mention about how to block it after the detection. Therefore, in this research, we propose and implement a method to detect and block DGA-based bot-infected computers by focusing on DNS domain name resolution process that do not require a pre-learning period or training data.

III. Proposed system

In this section, we describe the architecture and prototype implementation of the proposed system that detects and blocks the DNS domain name resolution from DGA-based bot-infected computers intends to search for its corresponding C&C servers in an organization network.

A. Overview

The proposed system is based on the two observations.

(1) NXDOMAIN responses are likely to be returned when the DGA-based bot-infected computers try to identify its C&C servers. This is the result of that attackers register only few DGA-based domain names for the C&C servers on the authoritative DNS servers, as described in Fig. 1.

(2) The same domain names will be generated and queried from multiple bot-infected computers if the same type of DGA is used in the bot program and the same seeds are used for the domain name generation.

Based on the above two observations, we consider the target threat model of this research is that when a computer is infected by other DGA-based bot-infected computers in the same network, it faces risk of being infected by the same bot. In this paper, we use the term "primarily infection" to refer the first bot infection occurred in a network and "secondarily infection" to refer the infection caused by the same bot from the primarily infected computer. Consequently, multiple computers in the network will query the same "DGA-based domain names" for identifying the C&C servers with a high possibility and the queries for those domain names that have not been registered in the authoritative DNS servers will result with NXDOMAIN responses. This is because that the same DGA with the same seeds will generate the same domain names. We apply this characteristic to the proposed system to detect and block the DNS domain name resolution from the DGA-based bot-infected computers.

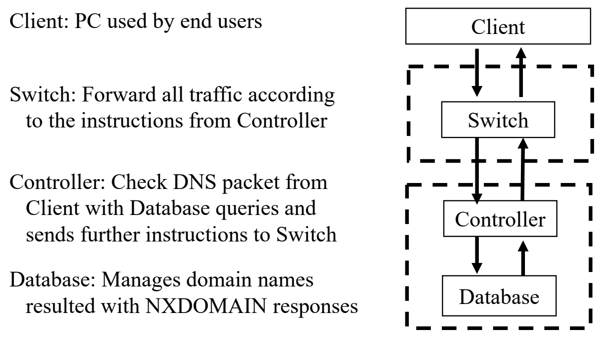
Client: PC used by end users
Switch: Forward all traffic according to the instructions from Controller
Controller: Check DNS packet from Client with Database queries and sends further instructions to Switch
Database: Manages domain names resulted with NXDOMAIN responses
Client
Switch
Controller
Database

Fig. 3. System model of the proposed system

Figure 3 illustrates the overview of the system model of the proposed system. The proposed system consists of two parts, DNS packet switching part and DNS traffic control part. The switch forwards all traffic from the client based on the instructions of the controller. The controller mainly checks the DNS packets by collaboration with the database and sends instructions to the switch. The database manages domain names that resulted with NXDOMAIN responses. Considering the comprehensive control features for network traffic and the implementation simplicity, we use Software-Defined Networking (SDN) technology to achieve the designed functionalities of the proposed system.

B. SDN

The proposed system uses SDN to detect and block DNS traffic sent from DGA-based bot-infected computers. SDN is a software-based centralized control technology for dynamically controlling network traffic with changing the policies. OpenFlow protocol is one of the popular SDN implementations. OpenFlow protocol provides network traffic control with two functions, data and control planes. Data plane is a function for switching packets that dynamically updates the flow table and forwards data based on the instructions of the controller. On the other hand, control plane is a function for instructing the data plane, which performs network path control and calculation for data forwarding. Accordingly, by using SDN technology, it is expected that the DNS traffic can be dynamically controlled based on the instructions of the controller with the domain name information on the database.

C. The procedure of SDN controller

As a prerequisite, all network traffic will be transmitted to the SDN switch by default. In the proposed system, since we focus on DNS traffic, all other traffic from an internal computer will be passed through to the destination on the SDN switch. If the received packet is a DNS packet, then the SDN switch forwards it to the controller for the further instructions called "packet-in" process in SDN. Then the controller checks the "packet-in" DNS packet in the database to confirm whether it should be forwarded or dropped. Figure 4 shows the verification process of the controller in detail. When the controller receives a DNS query through the "packet-in" process, it performs domain name verification, updates the blacklist database and gives instructions to the switch. We assume a case in which multiple computers have been infected by the same type of DGA-based bot in an organizational network. In this case, the bot-infected computers will generate same DGA-based domain names for searching C&C servers. Therefore, if the DGA-based domain names have not been registered in the authoritative DNS servers, the queries will get NXDOMAIN responses and the queried domain names will be stored in the blacklist database. Consequently, when the controller checks the queried domain names in the database the internal computers sent the same DNS queries will be detected and the further actions will be blocked.

There are two steps in the domain name verification. The first step is that the controller searches the queried domain name from the registered records in the blacklist database.

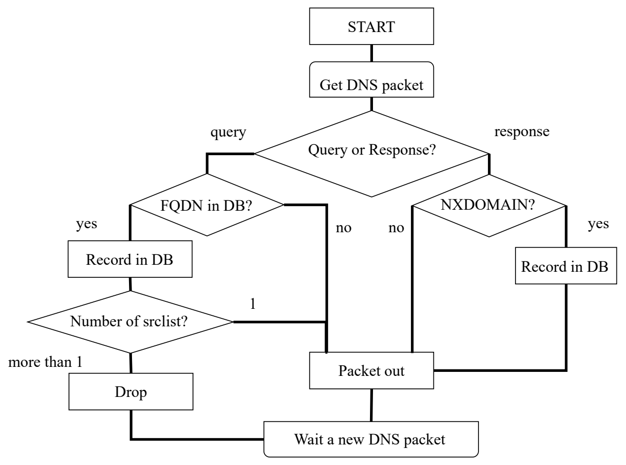

Fig. 4. Controller's procedure

If it does not exist, it is returned as a benign packet to the SDN switch that makes the DNS packet pass through although the DNS query may be from a primarily infected computer. However, if a record of the same domain name exists in the blacklist database, the controller adds the source IP address to the database and sends a control instruction to the switch to drop the DNS packets sent from the source IP address. The second step is that the controller checks the number of source IP addresses of the corresponding records in the blacklist database. If the number of the source IP addresses is one, it identifies that it is a computer configuration error or typing error and returns it to the switch as a benign packet. On the other hand, if the number of the source IP addresses is larger than one, it is unlikely that multiple users make same typos in a short period and there is a high probability that the bots are using the same type of DGA for searching the C&C servers. Therefore, the controller sends instructions to the switch to drop all packets sent from the source IP address.

The proposed system also includes an operation based on the received DNS responses in order to update the blacklist database dynamically. If the controller receives an NXDOMAIN response for a DNS query, the corresponding queried domain name with its relevant information will be added to the blacklist database. The main fields of the record in the blacklist database are described in the following. The number in parentheses indicates the size of the field in Bytes.

Fields achieved from the DNS response:

- `r_fqdn` bigint(20): the queried domain name
- `q_type` varchar(1000): the queried record type
- `r_time` bigint(20): the time when the controller receives the DNS packet
- `r_ns` varchar(1000): the zone name of the authoritative DNS server that authoritative for the queried domain name
- `r_nsttl` bigint(20): the negative cache time
- `srclist` int(11): the source IP addresses of the query

Fields achieved from the DNS query:

- `srclist`: the source IP addresses of the DNS query packets to which the `r_fqdn` and `q_type` are matched.

D. Implementation

Table I shows the specifications of the machines used for the prototype implementation. We created five virtual machines in the same network segment using KVM, a virtualization module, in one physical machine. Two of them were installed as a DNS full resolver and an authoritative DNS server, another two of them as client computers and the last one of them as an SDN controller and a blacklist database. In addition, we installed Open vSwitch (OVS) [12] as an SDN switch on the Host machine. For the controller, we adopted Ryu [13], which is a framework for developing the API of the controller part to realize the SDN-based network traffic control. According to the procedure described in Sect. III-C, we developed a Python program to analyze the DNS packets in collaboration with Ryu.

TABLE I
SPECIFICATIONS OF TESTING ENVIRONMENT

Component	OS	Software	CPU	RAM
Host machine	CentOS 7.7.1908	Open vSwitch (2.11)	Intel Xeon E5-1620 v2	16GB
Client1 (KVM)	7.6.1810	Original program	1 core	1GB
Client2 (KVM)	7.6.1810	Original program	1 core	1GB
DNS resolver (KVM)	7.6.1810	bind 9.11.4-P2	1 core	1GB
DNS authoritative server (KVM)	7.6.1810	bind 9.11.4-P2	1 core	1GB
Controller (KVM)	7.6.1810	ryu 4.32 MariDB 5.5.6	4 cores	2GB

In addition, in order to evaluate the functionalities of the proposed system in the experimental network environment, we also created a bot-like program by Python, i.e., only the DGA function of a bot as illustrated in (2) of Fig. 1 was implemented in the program. We implemented a program that generates DGA-based domain names by using the "date" as a seed and queries the domain name to the DNS resolver. This program updates the date seed, generates domain names and queries the domain name every day for a specified period from the specified date. In this system, the criteria for bot detection are deeply related to DGA-based domain name generation and its query. Therefore, in the evaluations, we can consider that the behavior of the created bot-like program works as a DGA-based bot-infected computer as we expected.

IV. Evaluation and Results

In this section, we describe the evaluation on the implemented prototype system and the results. First, we measured the process time of the DNS domain name resolution on the prototype system. Then we conducted feature evaluation to verify the functionalities of the proposed system.

A. Process time of the DNS domain name resolution

We measured the process time of the DNS domain name resolution in three different systems and the detailed information is shown in Tab. II. We compared the proposed system, which is described as "Packet-in with DB" in the table, with two other systems, Switching only (which do not use SDN controller), and Packet-in without DB (which is a normal SDN network). We used dnsperf [14], a tool for measuring DNS server performance, for the process time measurement. Specifically, we sent queries from the client to the DNS resolver for two different domain names which results with NOERROR and NXDOMAIN responses respectively and measured the process time on the three systems.

TABLE II
SYSTEMS USED FOR PROCESS TIME MEASUREMENTS

System	Switch-controller relationship	Extra process at the controller
Switching only	-	-
Packet-in w/o DB	Forward DNS packets to controller	-
Packet-in with DB (Proposed system)	Forward DNS packets to controller	Domain name check using DB

The results of the process time measurements are shown in Tab. III. It should be noted that except the system "Switching only", the DNS packets will be sent to the controller every time the "packet-in" event occurs. As a result, the Queries Per Second (QPS) of domain name resolution for "domain name with NOERROR response (NOERROR domain)" dropped from about 5,000 QPS to about 400 QPS. While when the proposed system processes an "NXDOMAIN domain", the average latency is about 3 seconds due to the time required to process the domain name verification in the blacklist database. This is larger than 2 seconds, which is the timeout limit of the "nslookup" tool used for the name resolution in operating systems. However, we consider that the performance of the DNS domain name resolution on the proposed system can be improved by tuning up the blacklist database operations.

TABLE III
RESULTS OF PROCESS TIME EVALUATION

System	Response message	Run time [s]	Query per seconds	Average latency [ms]	Latency stdDev [ms]
Switching only	NOERROR	1.523	5,254	18.92	4.310
Packet-in w/o DB	NOERROR	19.41	412.2	241.6	9.375
Packet-in with DB	NOERROR	21.39	374.1	262.2	15.62
Switching only	NXDOMAIN	1.516	5,277	18.81	5.165
Packet-in w/o DB	NXDOMAIN	19.04	420.2	237.0	16.21
Packet-in with DB	NXDOMAIN	226.0	35.40	2,810	378.7

B. Feature evaluation

1) Overview: In order to reproduce the various situations in an organizational network, we used several types of DGAs and scenarios in the experiment. Two scenarios were considered in the experiment and the outline is shown in Fig. 5.

- Experiment 1: Two bot-infected computers exist already.
- Experiment 2: One computer is infected in the primarily infection and another one is infected in the secondarily infection by the same DGA-based bot. This means there is some interval time exists between the two infections.

The above two scenarios were created with the following two factors related to the functions of the proposed system. Since these factors have a significant impact on the detection thus, we divided the scenarios based on the factors.

(a) whether NXDOMAIN information has been already registered in the database as prior information.
(b) whether multiple bot-infected computers are active.

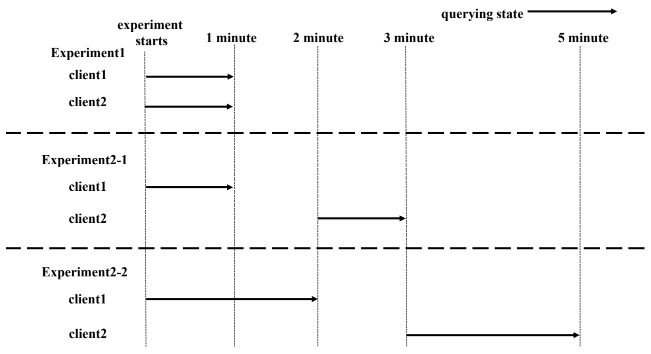

Fig. 5. Illustration of feature evaluation scenarios

In the experiments, one day of the actual time is shortened to one minute to shorten the evaluation time. Therefore, in Fig. 5, the number of elapsed minutes after the start of the experiment corresponds to the number of days elapsed in the actual time. In other words, the DGA-based bot program generates domain names from the date seed and queries it every minute for a specified period. In the Experiment 1, two bot-infected computers simultaneously generate and query the domain names of the same date seed for one minute. In the Experiment 2, the primarily and secondarily bot-infected computers query at different time. The amount of NXDOMAIN information in the blacklist database of the proposed system varies with the length of time. Therefore, in the Experiment 2 (2-1 and 2-2), we verified the case where each bot-infected computer queries for 1 or 2 minutes. We also set one minute of idling time for the secondarily infected computer. In the experiment, 23 TDD-type DGAs were used out of publicly available Python programs [15] introduced to generate similar DGA-based domain names by reverse-engineering the DGA part of the existing bot programs. These bot-like programs have fixed the input seed for day 1 at 2020-01-06 within the parameters.

2) Results: The number and names of detected DGAs are shown in Tab. IV. In the Experiment 1, the DNS domain name resolution from bot-infected computers was detected and blocked against all types of DGAs. While in the Experiment 2-1 and 2-2, 8 and 11 DGAs were detected and blocked respectively. In the Experiment 1, multiple bot-infected computers queried the same DGA-based domain names thus the proposed system was able to detect them in all the prepared DGAs. On the other hand, in the Experiments 2-1 and 2-2, the proposed system was able to detect only about half the DGAs prepared. The DGAs failed to be detected had different date seeds in the primarily and secondarily infected computers thus, the generated DGA-based domain names did not overlap. As a result, the domain name registered in the blacklist database had only one type of IP address and the queried computer was not detected as a bot-infected computer.

In addition, we also analyzed the detected DGAs regarding the domain name generation interval and the detailed results are shown in Tab. V. The results show that the period of

TABLE IV
RESULTS OF FEATURE EVALUATION

Experiment	# of detected DGAs	Name of DGA
Experiment 1	23	all
Experiment 2-1	8	Qakbot, Sisron, Mydoom, Symni, Kraken_v2, Tempedreve, Nymaim2, Pykspa_improved
Experiment 2-2	11	Bots found in Experiment 2-1 (Qakbot, Sisron, Mydoom, Symni, Kraken_v2, Tempedreve, Nymaim2, Pykspa_improved) and Pykspa_precursor, Murofet_v1, Murofet_v3

domain name generation was adjusted based on the DGA types. There are three types of period adjustment: (1) seed processing, (2) sliding window and (3) accumulation. Table V shows the period along with the classification for the detected DGAs (bots) that are considered in the form (1) and (2).

TABLE V
CLASSIFICATION OF SEED TYPES

Name of DGA	Type	Cycle [day]	Name of DGA	Type	Cycle [day]
Pykspa_improved	1	20	Symni	1	15
Pykspa_precursor	1	2	Kraken_v2	1	7
Murofet_v1	1	7	Nymaim2	2	11
Murofet_v3	1	7	Sisron	2	9
Qakbot	1	10			

Type: (1: Seed proceeding, 2: Sliding window)

(1) In the seed processing type, the number of generated DGA-based domain names is constant in a certain period. This is because the seed is truncated by a constant, for example in weeks, before it is generated by the random number generator. Therefore, if UNIX TIMESTAMP is used for the seed calculation, it is truncated by the specified number of $\text{days}*24*3600$ and is delivered to the random number generator. As a result, same domain names will be generated for the specified number of days.

(2) In the sliding window type, multiple seeds are used. This type of DGA generates N new domain names with a single seed (date) then queries them within a certain period (M days before). That means that a total of $N+M*N$ queries per day will be performed. Therefore, there are N domain names in common for a maximum of $M+1$ days.

(3) Unlike (2), there is no limit to a certain number of M days in the accumulation form. This form queries all newly generated domain names by the hardcoded date B, in addition to the N newly generated domain names on the seed date A. Therefore, compared to (2), the number of domain names queried is $N*(A-B+1)$ and it increases every day.

As a result, N new DGA-based domain names are queried every cycle with TID type and once generated domain names were not deleted but queried every time the accumulation form DGA was executed. Based on this analysis and the difference in the number of detection types in Experiments 2-1 and 2-2, we can confirm that the proposed system can detect more diverse DGA-based bot-infected computers by extending the monitoring period.

V. SUMMARY

In this paper, we proposed a method to detect and block DNS domain name resolution from DGA-based bot-infected computers for searching the C&C servers by focusing on the DNS queries resulting with NXDOMAIN responses. The queried DGA-based domain names caused NXDOMAIN responses will be managed in the blacklist database for the detection and blocking the further actions of DGA-based bot-infected computers. We implemented a prototype system and evaluated the features and domain name resolution performance in a local SDN experimental network. Based on the evaluated results, we confirmed that the proposed system was capable of detecting all DGA-based bots using TDD-type seeds when multiple computers are primarily infected by same type of DGA-based bots.

We also found that when the bot infection was slower than the DGA-based domain name generation cycle the proposed system could not detect the DNS traffic after the infection. In order to deal with this case, it is necessary to develop a detection mechanism that not only depends on the behavior of multiple computers but also on the behavior of a single computer, such as the researches in Sect. II-B.

REFERENCES

[1] JPCERT/CC, "Alert regarding Emotet malware infection," https://www.jpcert.or.jp/at/2019/at190044.html, Nov. 2019.
[2] A.K. Sood, and S. Zeadally, "A taxonomy of domain-generation algorithms," IEEE Security & Privacy, vol. 14, no. 4, pp. 46–53, July-Aug. 2016.
[3] M. Feily, et al., "A survey of botnet and botnet detection," Proc. IEEE Int'l Conf. Emerging Security Information, Systems and Technologies, Athens, Glyfada, June 2009, pp. 268–273.
[4] H. Ichise, Y. Jin, and K. Iida, "Analysis of via-resolver DNS TXT queries and detection possibility of botnet communications," Proc. IEEE Pacific Rim Conf. Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Aug. 2015, pp. 216–221.
[5] 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.
[6] H. Ichise, Y. Jin, K. Iida, and Y. Takai, "NS record history based abnormal DNS traffic detection considering adaptive botnet communication blocking," IPSJ J. Information Processing, vol. 28, pp. 112-122, Feb. 2020.
[7] Y. Liu, "Now Mirai has DGA feature built in," https://blog.netlab.360.com/new-mirai-variant-with-dga/, Dec. 2016.
[8] D. Plohmann, et al., "A comprehensive measurement study of domain generating malware," Proc. USENIX Security Symp., Austin, USA, Aug. 2016, pp. 263–278.
[9] S. Yadav, et al., "Detecting algorithmically generated malicious domain names," Proc. ACM Annual Conf. Internet Measurement (IMC'10), Melbourne, Australia, Nov. 2010, pp. 48–61.
[10] M. Antonakakis, et al., "From throw-away traffic to bots: Detecting the rise of DGA-based malware," Proc. USENIX Security Symp., Bellevue, USA, Aug. 2012, pp. 491–506.
[11] Y. Li, et al., "A machine learning framework for domain generation algorithm-based malware detection," IEEE Access, vol. 7, pp. 32765–32782, Jan. 2019.
[12] "OpenvSwitch," https://www.openvswitch.org/, Accessed at Jan. 2020.
[13] "Ryu," https://github.com/osrg/ryu, Accessed at Jan. 2020.
[14] "DNS Performance Analytics and Comparison," https://www.dnsperf.com, Accessed at Jan. 2020.
[15] "Domain generation algorithm," https://github.com/baderj/domain_generation_algorithms, Accessed at Jan. 2020.