Detection and Blocking of Anomaly DNS Traffic by Analyzing Achieved NS Record History

Hikaru Ichise*, Yong Jin†, Katsuyoshi Iida‡ and Yoshiaki Takai‡
* Graduate School of Information Science and Technology, Hokkaido University, Japan
and Technical Department,Tokyo Institute of Technology, Japan
E-mail: hichise@nap.gsic.titech.ac.jp
† Global Scientific Information and Computing Center, Tokyo Institute of Technology, Japan
E-mail: yongj@gsic.titech.ac.jp
‡ Information Initiative Center, Hokkaido University, Japan
E-mail: {iida,takai}@iic.hokudai.ac.jp

***Abstract*—DNS (Domain Name System)-based 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, it is observed in some malware that DNS queries are sent to C&C servers using IP address directly without obtaining the corresponding NS records. In this paper, we propose a novel mechanism to detect and block anomaly DNS traffic by analyzing the achieved NS record history in an organization network. In the proposed mechanism, all DNS traffic of an organization netwok will be captured and analyzed in order to extract the legitimate NS (Name Server) records and the corresponding glue A records (the IP address(es) of a name server) which will be stored in a white list database. Then all the outgoing DNS query packets will be checked and those destined to the IP addresses that not included in the white list will be blocked as anomaly DNS traffic. We have implemented a prototype system and evaluated the functionalities in an SDN-based experimental network. The results show that the prototype system works as expected and the proposed mechanism is capable of detecting and blocking some specific types of suspicious DNS traffic.**

I. Introduction

Botnet, a malicious logical network of cyber attackers, has become a critical security threat in cyberspace [3], [4]. 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 [5], [6]. 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, we 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 regarding to the above workflow, in this research we target the botnet communication as a means to analyze and detect botnet-based cyber attacks.

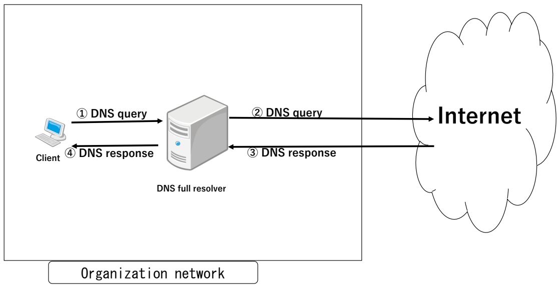


Fig. 1. Legitimate use case of DNS

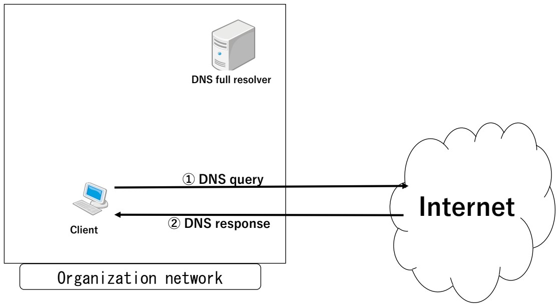


Fig. 2. Suspicious use case of DNS

Many recent reports indicate that DNS protocol [7], [8] is used in botnet communications [9], [10], [11]. DNS protocol is mainly used for name resolution, which is translating hostnames to IP addresses in the Internet. However, the increase of Internet services has led to wide uses of some minor records such as DNS TXT record. In [12], 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 [13], [14], Ichise et

Earlier version of this paper has been presented in [1], [2]. The major contribution of this paper from [1], [2] is Sect. V.A and V.B.

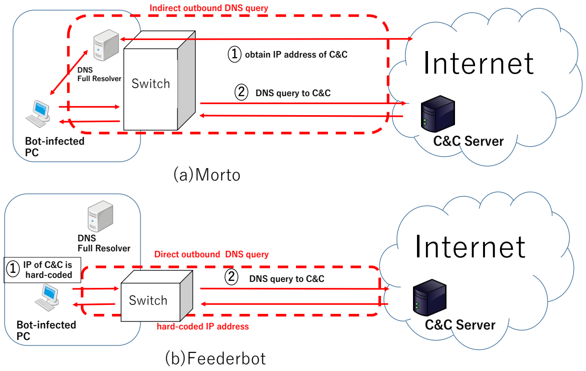
al. analyzed DNS packet traces to differentiate the legitimate and suspicious uses of TXT records. Consequently, DNS packets which are currently considered to be secure network traffic have also become a target of being monitored communications since network administrators cannot simply block all DNS traffic. Thus, an effective detecting and blocking solution for DNS-based botnet communications is needed.

Figure 1 shows general name resolution process with a deployed DNS full resolver in an organization network. A client PC firstly sends a DNS query to the DNS full resolver for requesting name resolution. Then the DNS full resolver performs name resolution on behalf of the client by querying the corresponding authoritative DNS servers in the Internet and finally replies back the DNS response to the client. In the name resolution process, the DNS full resolver achieves the corresponding NS records and glue A records of the authoritative DNS servers in prior to sending DNS queries to them. Here, NS record indicates a host name of the 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 all cases, the internal clients relies the DNS name resolution on the DNS full resolvers of the organization network. On the other hand, botnet communications may not follow this procedure. Figure 2 shows a typical anomalous DNS traffic such as a type of botnet communication using DNS protocol. In this example, a client PC sends a DNS query to the Internet directly without using the DNS full resolver of the organization network. We call this type of DNS communication as direct outbound DNS query and response (Q&R) [15]. However, there also exist some exceptions, in which this type of direct outbound DNS Q&R can be used by legitimate use cases. For example, in this case of using public DNS servers such as Google Public DNS, a client may send direct outbound DNS queries to the public DNS servers without using the DNS full resolvers of the organization network.

In summary, DNS name resolution is performed by DNS full resolvers in almost all legitimate use cases, whereas it is not in suspicious use cases. Based on this observation, we propose a detecting and blocking method of anomalous DNS traffic. Our key idea is that name resolution obtains NS and glue A records of the corresponding authoritative name servers in 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 paper, we focus on detecting and blocking the anomalous DNS traffic by analyzing the achieved NS records history.

II. Overview of DNS-based botnet communication and related work

As stated in the Introduction, the objective of our research is to construct a system for detecting and blocking DNS-based botnet communication. 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.



(a)Morto

(b)Feederbot

Fig. 3. Direct & indirect outbound DNS queries

A. Three types of DNS-based botnet communication

In an organization network, one or more DNS full resolvers will be set up for providing DNS name resolution service to the internal computers. The internal computers send DNS queries to the DNS full resolvers and the DNS resolvers perform the name resolutions on behalf of the internal computers and finally reply back the DNS responses to them. As we mentioned before, once an internal computer is infected by some bot programs, the infected computer needs to contact with its C&C servers which is called botnet communication.

We 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 [16], the authors have reported that the uses of DNS TXT records in botnet communication using via-resolver DNS query. Next, botnet communication using indirect outbound DNS query is detected in a bot program named Morto [17]. Figure 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 C&C server directly. On the other hand, a bot program named Feederbot never uses DNS full resolvers, which is called as direct outbound DNS query [15]. Figure 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 authoritative NS record before sending DNS queries to the C&C servers. Therefore we consider that if we could collect obtained authoritative NS as well as glue A records and check the destination IP addresses of all the DNS queries in the organization network, it will be possible to detect and block these types of DNS-based botnet communication.

B. Related work

Since DNS-based botnet communication might be used widely there were some researches in the literature. In [18],

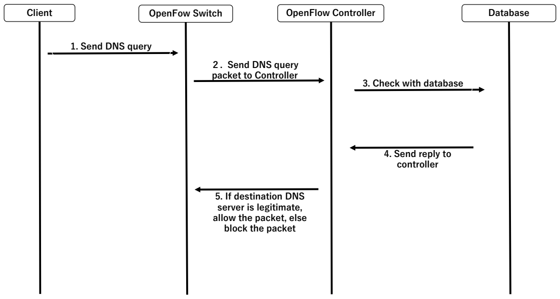

Client
OpenFow Switch
OpenFlow Controller
Database
1. Send DNS query
2． Send DNS query packet to Controller
3. Check with database
4. Send reply to controller
5. If destination DNS server is legitimate, allow the packet, else block the packet

Fig. 4. Overview of the workflow in the proposed system

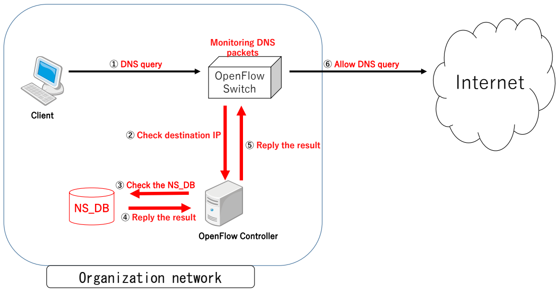


Fig. 5. Detail Procedure of the Proposed Method

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 DNS query and response packets with focusing on Domain Generation Algorithms. In traffic analysis, they analyzed over time in DNS traffic. In [15], the authors analyzed 14 million DNS TXT queries and found the bot program named “Feederbot”. 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, we 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.

III. Proposed system

In order to detect and block DNS-based botnet communications, we first construct a database of white listed DNS NS and glue A records. This list includes the DNS NS records of domain names achieved from all received legitimate DNS responses such as domain names “.com” and “.net” and their corresponding glue A records. We 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 we consider them as legitimate, while to those not listed in the white list we will drop the packets and log them down for further investigations. We use SDN (Software Defined Network) 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, the controller inquiries 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 shows the brief workflow of the proposed system.

A. Construction of NS record history database

First, we construct the NS_DB from the corresponding DNS queries and responses. One of the simple methods is to obtain all DNS traffic from an organization network and pick out legitimate DNS NS records as well as the corresponding glue A records. For the simplicity, we create query_DB first for storing all legitimate DNS queries and using all the achieved legitimate DNS responses to the queries we construct the NS_DB. We 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 DNS query the analysis program performs the procedure for DNS query otherwise for DNS response. In the DNS query procedure, 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 that the DNS responses only have NS records, only the NS records will be registered to the NS_DB. Then the analysis program continuously monitoring DNS traffic and 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.

B. System architecture

Terminologies used in the proposed system:

- Query Fully Qualified Domain Name (Query FQDN): The hostname or domain name that the clients query.
- Query destination IP Address: The destination IP address to which the clients send DNS queries.
- Database: For storing the legitimate DNS NS and glue A records as well as legitimate public DNS servers.
- Owner name (zone): The domain name for which a DNS server is authoritative.

Figure 5 shows a simple system architecture of the proposed method with the basic procedure when a client sends a direct outbound DNS query to the Internet.

1) The client sends a DNS query to the Internet and all the DNS query packets from the client are monitored on the OpenFlow Switch.
2) The OpenFlow Switch checks the destination IP address of the DNS query packet with the OpenFlow Controller

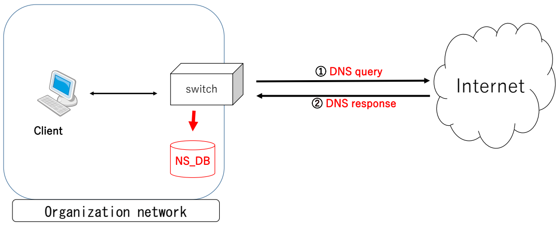


Fig. 6. Network topology for obtained data

if there is no flow entry for the destination IP address of the DNS query packet.

3) The OpenFlow Controller inquires the database for the destination IP address.
4) The database replies the check result to the OpenFlow Controller.
5) The OpenFlow Controller makes decision based on the check results. If the destination IP address of the DNS query packet is in the database the OpenFlow Controller passes the DNS query packet, otherwise drops it.

With the above procedure, the proposed system can be expected to detect and block the target (some specific types) DNS-based botnet communications.

IV. Implementation

A. NS record history database

We implemented a program for constructing the query_DB and NS_DB databases. In the implementation, we used python as the program language, dpkt module for decoding pcap files and MariaDB as the database system.

Figure 6 illustrates the brief network topology we used for obtaining the DNS traffic. All DNS queries (the arrows numbered 1 in Fig 6) and DNS responses (the arrows numbered 2 in Fig 6) from the border router were captured 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. Table I shows an example of the entry in the NS_DB. Note that it is important to delete this table when nsttl or glueAttl expires. Moreover, if the DNS response only includes NS records they will be registered without glue A records. After that, if the glue A records are received continuously within two seconds the glue A records will be added in the NS_DB, otherwise, the NS records will be deleted. Some specific IP addresses such as those of public DNS servers and those of update servers for anti-virus software so that they will also be added to the NS_DB.

B. Detecting and blocking features

We implemented the detecting and blocking features using SDN technology, specifically using OpenFlow solutions. We chose Open vSwitch, which is a free virtual switch as the OpenFlow switch and Ryu program as the OpenFlow controller. For the host and guest operating systems, we chose a Linux operating system: CentOS 7.4.

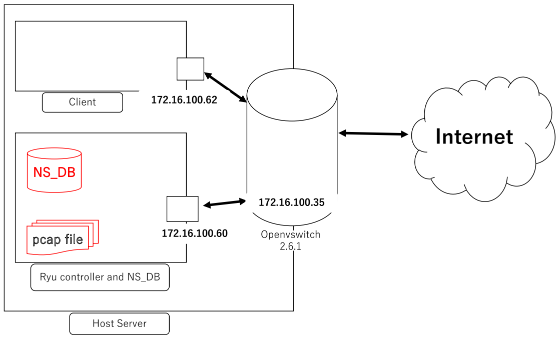


Fig. 7. Network topology for evaluation

V. Evaluation

We evaluated the implemented prototype system in a local experimental network in order to check the features of the proposed system. As this is a prototype, there was not much stress test and we mainly focused on the functionality of the controller program to check if the system could detect and block malicious DNS traffic. The evaluations include database construction part and malicious DNS traffic detection and blocking part. We describe the two parts in the following Sect. V-A and Sect. V-B.

A. Network environment

To conduct the evaluation of the prototype system, we have set up a local experimental network environment consists of an Open vSwitch, a Ryu OpenFlow controller and a client as shown in Fig. 7. The Open vSwitch was installed on a host machine and the Ryu controller and the client were installed in KVMs in the host machine. The Ryu controller and the client were configured to be connected to the Open vSwitch and they could only communicate with the Internet through the Open vSwitch. For the database system we used to store the NS record history, we constructed it on the Ryu KVM. Consequently, all traffic from the client will pass through the Open vSwitch and thus the Ryu controller can check the destination IP address as well as can decide whether pass it or not. In our previous study [19], we obtained many pcap files of DNS traffic from the border router of our campus network. In the evaluation, we used the pcap files for constructing the NS record history database. A pcap file of DNS traffic for about two hours approximately has size of 200MB and we used it in the evaluation. It took about one hour to analyze the pcap file and to store the NS records history in the database. As as result, 18,079 legitimate NS records were picked from the pcap file and be stored in NS_DB.

B. Feature evaluation

After constructing the NS records history database in the experimental network environment, we evaluated the malicious NS traffic detection and blocking feature of the proposed method which is mainly realized by the Ryu controller program. Specifically, we attempt to check destination IP address

TABLE I
COLUMN 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 II
RESULT OF FEATURE EVALUATION

Run the command in client	Result of checking destination IP address by NS_DB in controller	The behavior of Open vSwitch
nslookup www.google.com 8.8.8.8	8.8.8.8 is unknown	block
nslookup www.google.com 8.8.4.4	8.8.4.4 is registered glueA record	pass
nslookup www.verisign.net 192.26.92.31	192.26.92.31 is registered glueA record	pass

of DNS query sent from the client with the glue A records stored in the NS_DB in the Ryu controller and Open vSwitch will be instructed by the Ryu controller to block the DNS query properly in case of no glue A record in the NS_DB and pass through when it has hit in the NS_DB.

We sent DNS query from the client (172.16.100.62) to the Internet using command described in table II to test the Ryu program. When the Open vSwitch receives the DNS query packet for the first time it will be forwarded to the Ryu controller via the "packet_in" process since there is no entries for the DNS query packet from the client in the flow table. Then the Ryu program monitors the DNS query packet and checks its destination IP address with the with glue A record stored in the NS_DB. If the destination IP address is stored as a glue A record in the NS_DB, the Ryu program displays "<destination IP address>is registered glue A record and pass." message and passes the DNS query packet otherwise displays "<destination IP address>is unknown and block." message and blocks the DNS query packet. We show the evaluation results in table II. We 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 is not registered in the NS_DB, 8.8.4.4 which is another open DNS resolver provided by Google and is registered NS_DB, and 192.26.92.31 which is an authoritative DNS server of the domain name "verisign.net" and is registered in the NS_DB. This result showed that the prototype worked well as we expected.

VI. CONCLUSION

To conclude, the purpose of this paper is to propose a solution to detect and control DNS based botnet communications by monitoring direct outbound DNS query. We have set up a test environment and implemented a prototype system based on our proposed method using python based SDN solution Ryu and also evaluated on a local experimental network. The evaluation results showed that the proposed system can possibly detect and block malicious DNS traffic. In the future work, we attempt to use this NS record history database in actual environment. In addition, we will evaluate to detect DNS-based botnet communication using NS_DB in our campus network.

REFERENCES

[1] 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, Mar. 2016.

[2] H. Ichise, Y. Jin, and K. Iida, "Design and implementation of NS record history database for detecting DNS-based botnet communication," *IEICE Tech. Rep.*, vol. 117, no. 299, IA2017-31, pp. 7–11, Nov. 2017.

[3] 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

[4] 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

[5] 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.

[6] McAfee Labs, "Threats report," https://www.mcafee.com/tr/resources/reports/rp-quarterly-threats-mar-2018.pdf, March 2018.

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

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

[9] 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

[10] 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.

[11] 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

[12] 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.

[13] 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

[14] 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

[15] 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

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

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

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

[19] 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