EPiC Series in Computing
Volume 58, 2019, Pages 137–143
Proceedings of 34th International Conference on Computers and Their Applications

A Multi-Stage Detection Technique for DNS-Tunneled Botnets

Tirthankar Ghosh[1], Eman El-Sheikh[1] and Wasseem Jammal[2]
[1]University of West Florida, Pensacola, FL, U.S.A.
[2]St. Cloud State University, St Cloud, MN, U.S.A.
tghosh@uwf.edu, eelsheikh@uwf.edu

Abstract

Botnet communications are obfuscated within legitimate network protocols to avoid detection and remediation. Domain Name Service (DNS) is a protocol of choice to hide communication with Command & Control (C&C) servers, where botmasters tunnel these communications within DNS request and response. Since botnet communications are characterized by different features, botmasters may evade detection methods by modifying some of these features. This paper proposes a multi-staged detection approach for Domain Generation Algorithm (DGA) using domain fluxing, Fast Flux Service Network (FFSN), and encrypted DNS tunneled-based botnets using BRO Network Security Monitor. This approach is able to detect DNS-tunneled botnet communications by analyzing different techniques used to find C&C servers, and also using signature matching technique to detect DNS-tunneled SSH handshake between bots and C&C servers.

1 Introduction

Over the years, botmasters have invented various techniques to obfuscate botnet communications with C&C servers. Since Domain Name Service (DNS) is one of the very few protocols that is allowed to pass network perimeters without much question, it has become a protocol of choice for botnet communications. In addition, different techniques, like Domain Generation Algorithm (DGA) and Fast Flux (FF), have been used by botmasters to make detection and remediation harder.

Domain generation algorithms (DGA) are used to periodically generate a large number of domain names that can be used to resolve IP addresses of their Command and Control (C&C) servers. "DGAs allow the algorithmic generation of a large set of pseudorandom domain names using a pre-calculated seed value known to the attacker. The code for generating the seed value is embedded in the malware binary to generate domains dynamically" [1]. This technique gained traction from the release of the Conficker series of malware, which generated 50,000 domain names per day, requiring malware analysts to pre-register 50,000 domains every day to avoid infection. While DGA generates large number of domains per day, "fast-flux is a technique to cycle the mappings of domain names to IP addresses of hosts participating in a botnet, often with short lifetime mappings" [2], making detection and remediation extremely hard. One effective remediation technique is to shut down the domain name at the registrar level, but that takes time and cooperation from all registrars.

In addition, bots tend to initiate a secure shell (SSH) communication, tunneled within DNS, to establish an encrypted communication channel with their C&C servers. This obfuscation technique adds further complexity to the already-difficult detection and remediation solutions for botnets.

In this paper, we propose a multi-staged technique to detect botnet communications using DGA, FF, and DNS-tunneling. We used the Bro Network Security Monitor (NSM) [3] to create rules based on network anomaly and signature to detect such behavior.

2 Related Work

Dynamic domain generation, fluxing, and tunneling techniques have been used by different malware families to avoid detection and complicate mitigation efforts. Research communities have proposed many different approaches and mechanisms to detect botnet communications, most of which are effective for specific types of botnet. Krmíček [4] examined the NetFlow1 of DNS IP traffic and its relation to the botnet presence in the monitored network. The author studied the DNS behavior of known malicious and benign domains based on features identified by Bilge et al. [5]. Since NetFlow inspects only packet headers, not the entire packet payload, Krmíček concluded that "using NetFlow data solely, for the purpose of botnet detection is not possible" [4], and he mentioned that extracting important information from the packet payload is the most promising approach for botnet detection.

Dietrich et al. [6] are the first to document DNS-based botnet C&C traffic. They presented a technique for DNS-based C&C traffic detection and another technique for malware sample classification based on their behavior. Their work is based on the high entropy of C&C messages generated by Feederbots; they utilized the fact that encrypted or compressed messages have high entropy. Non uniform distribution of some resource records such as IP addresses will be a limitation of their study (e.g., reserved addresses, such as private addresses or multicast addresses, might rarely show up in Internet DNS traffic), whereas other addresses, such as popular websites, might appear more often in DNS query results. Another limitation is that very short DNS messages may not have the byte entropy sufficient to produce the desired result.

Lysenko et al. [7] proposed a DNS-based anti-evasion technique for botnet detection. Their technique is based on a cluster analysis of the features obtained from the payload of incoming DNS messages. The method uses the semi-supervised fuzzy c-means clustering.

Jin et. al. [8] designed a botnet communication detection method by collecting authoritative NS records and their IP addresses, as well as monitoring direct outbound DNS queries. Their method is based on storing NS records with corresponding IP addresses of valid query response pairs, IP addresses of public DNS servers, and ISP specified DNS servers in a NS-IP database. Any destination IP address not included in the previously archived Name Server (NS) records is considered suspicious and should be investigated. In this way, "all unusual domain name resolution that uses direct outbound DNS query can be monitored" [8].

Caglayan et al. [9] presented the first empirical study of detecting and classifying Fast Flux Service Networks (FFSNs) in real time. Their approach uses active and passive sensors derived from DNS monitoring and fusing the component sensors using a Bayesian classifier. The Fast Flux Monitor Architecture can detect single and double flux behavior in real time with acceptable false alarm rates.

Dabbagh [10] proposed a method for detecting IP ID and TTL covert channels. He proposed a method based on his observation that "operating systems choose initially a random number for the ID in the IP header and then increment it sequentially" [10]. Dabbagh concluded that a packet is suspicious if the new packet has an IP ID smaller than the previous packets. Also, he stated that "detecting TTL covert channel is based on the fact that the network is stable" [10]. Therefore, the receiver side should not observe many variations in the TTL values in the IP header of the packets that are coming from the same source.

Zhang et al. [11] made an initial attempt to investigate detection of encrypted communication. Since the encryption increases entropy, they presented two high-entropy classifiers, used one of them to enhance the BotHunter, and showed that BotHunter was able to detect encrypted bots.

Antonakakis et al. [12] presented a novel detection system, called Pleiades, which is able to detect machines within a monitored network that are compromised with DGA-based botnets. Pleiades monitors traffic below the local recursive DNS server and analyzes streams of unsuccessful DNS resolutions (Name Error or NXDomain Responses). Pleiades searches for relatively large clusters of NXDomains with similar syntactic features, and are queried by multiple, potentially compromised, machines during a specific epoch. There are two phases of detection: the first phase discovers the presence of DGA and the second classifies the discovered DGA and detects the C&C domain(s). Although their claim that Pleiades can achieve very high detection accuracy, one limitation of their evaluation method is "the exact enumeration of the number of infected hosts in the ISP network" [12]. Because the location of monitoring sensors is below the recursive DNS server, they can only obtain a lower bound estimate of infected hosts. For example, an IP address that generates DNS traffic may be a NAT, firewall, DNS server, or other device that behaves as a proxy. Also, noisy NXDomains may be generated to mislead the implementation of Pleiades.

Yadav and Reddy [13] proposed methodologies for utilizing failed domain names (NXDOMAIN) in the quest for rapid detection of a fluxing botnet's C&C server. They validated their method by detecting Conficker botnets and other anomalies with a false positive rate as low as 0.02%. Their technique can be applied at the edge of an autonomous system for real-time detection. Since their method is based on detecting botnets utilizing high entropy, botnet owners may alter the way domain names are created to evade their detection mechanism.

In this paper, we propose a multi-staged detection technique to detect DNS-tunneled botnet communications. The first stage utilizes NXDomain and Server Failure errors to detect rallying to C&C servers. In the second phase, the algorithm uses a detection technique relying on a signature matching based on encoded SSH handshakes within DNS tunnels.

3 Proposed Technique

This paper proposes a novel solution to detect DNS-tunneled botnets at different stages – during rallying stage when finding the C&C server (DGA and FFSN), and during transmitting data and controlling the bots (DNS Tunnel).

Currently, botnets implement DGA and/or fluxing techniques to avoid botnet detection and mitigation. The infected machine sends a high volume of DNS requests in order to find its C&C server. As a botmaster only registers one or a few domain names (previously known) to carry out the C&C communication, almost all DNS requests, generated by DGA, sent to find the C&C server will have unsuccessful resolutions (name error or NXDomain responses). Detection of DGA implementation was configured based on a threshold of NXDomain responses within an epoch. For example, if the infected machine sends more than 50 DNS requests within a thirty-minute epoch, and a specific percentage of these requests have unsuccessful resolutions, BRO NSM will detect the presence of DGA based botnet.

In FFSN, malicious domain name(s) that has/have very low time to live (TTL) (domain expiration time) forces the DNS systems to frequently refresh the resolution cache of the IP addresses associated with the domain(s). Although the DNS Server Failure can be related to issues other than fluxing implementations, these unsuccessful resolutions of very low TTL domain names (Server Failure) can be utilized for FFSN-based botnets detection. In other words, the detection of FFSN and DF implementation was configured based on a threshold of the "Server Failure" responses within an epoch.

The frequency of malicious DNS packets can be controlled by the botmaster to evade the detection threshold. In other words, in case the first stage fails to detect DGA or FFSN presence, another stage will run. The second stage inspects DNS payloads for DNS-encrypted tunnels based on SSH connections, also implemented with BRO NSM. The multi-staged approach is shown in Figure 1.

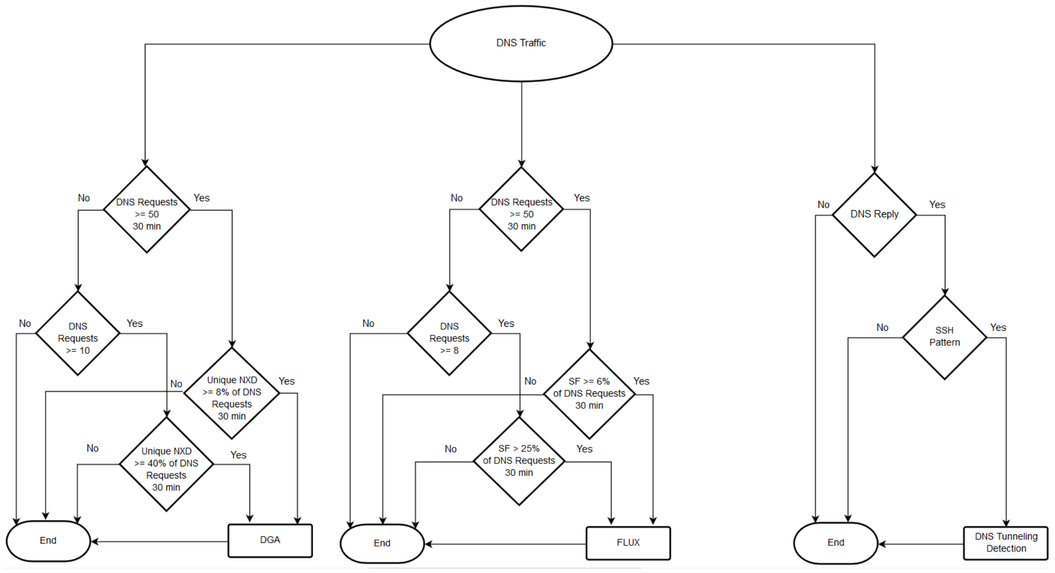

DNS Traffic
DNS Requests >= 50 30 min
No
Yes
DNS Requests >= 10
Unique NXD >= 8% of DNS Requests 30 min
Unique NXD >= 40% of DNS Requests 30 min
End
DGA
DNS Requests >= 8
SF >= 6% of DNS Requests 30 min
SF > 25% of DNS Requests 30 min
FLUX
DNS Reply
SSH Pattern
DNS Tunneling Detection

Figure 1: Multi-staged approach for botnet detection

The thresholds are based on data published by Brandhorst and Pras [14] on statistical analysis of domain name service traffic. According to the study, percentages of NXDomain errors and Server Failures were 8.74% and 1.28% respectively of the DNS queries.

In the first stage of botnet communication (rallying stage), a DGA-detection mechanism is applied every 30 minutes to find if total number of DNS requests is 10 or greater. In this case, two thresholds are used for the unique NXDomain errors: the first threshold, which is 8% of the total number of DNS requests, and used if the total number of DNS requests is 50 or more. The second threshold (40%) is used if the total number of DNS requests is between 10 and 50. The second threshold is utilized to eliminate false positive detections in idle systems. For example, an idle system running DGA has a higher percentage of unique NXDomain errors to the total number of DNS requests than the percentage in an active system running DGA. This threshold should be varied to analyze changes in false positive detection. In both thresholds, at least four unique NXDomain errors are required for DGA detection.

In fluxing detection, another method used for rallying, different thresholds are set. The fluxing detection mechanism is applied every 30 minutes if the total number of DNS requests is 8 or greater. In this mechanism, two thresholds are used for the Server Failure errors. If the total number of DNS requests is 50 or greater, the first threshold is used, which is 6% of the total number of DNS requests. The second threshold (26%) is used if the total number of DNS requests is between 8 and 50. Similar to the DGA mechanism, the second threshold is utilized to eliminate false-positive detection in idle systems implementing fluxing techniques, which should be varied for further analysis. In both thresholds, at least three Server Failure errors are required for fluxing detection.

Figure 1 depicts the first stage of the detection algorithm to detect DGA and FF as two parallel stages for simplicity of representation. However, during implementation, these two stages are combined as one stage, which is the first stage of our detection algorithm. The DNS threshold, which is 50 DNS requests every 30 minutes, is set below the lowest average from Brandhorst and Pras' study [14]. In other words, different threshold values are set for DGA and fluxing detection. These values need to be dynamically adjusted with the changing nature of communication.

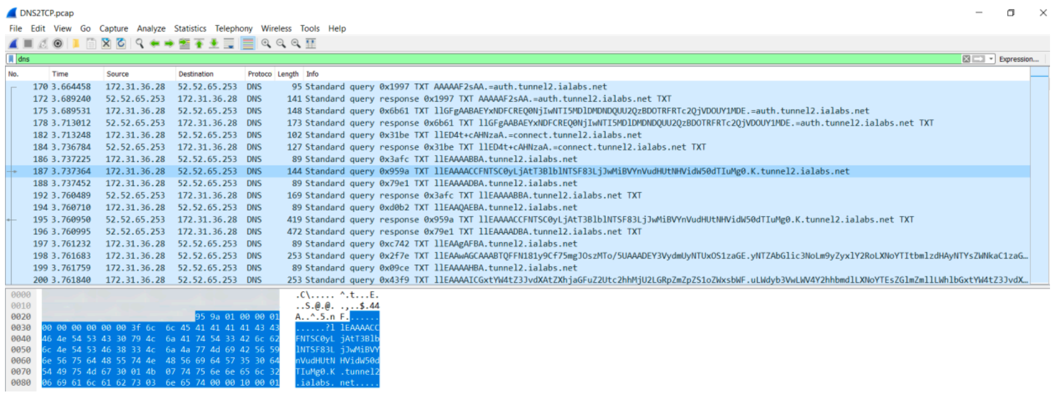
Bro Network Security Monitor (Bro NSM) [3] was used as the intrusion and anomaly detection platform to implement both phases of the detection technique. Detailed discussion of implementation and results are given in the next section.

4 Implementation and Results

We used Bro Network Security Monitor (Bro NSM) to implement the detection algorithm. Bro provides an anomaly detection platform with its own scripting language called Bro scripting with an event-driven approach. Two events were created for DGA detection: *dns_request* and *dns_query_reply*, and *Sumstats* framework was used to summarize the desired statistics. Similarly, two events, *dns_request* and *dns_rejected* were created for Fast Flux detection.

The first phase of the detection technique was tested on the dataset collected from the Stratosphere Lab, which is a part of the Malware Capture Facility Project at CVUT University, Prague, Czech Republic [15]. The lab has a significant dataset of malware traffic captures, including different types of botnets. These datasets were used for DGA and FFSN detection. When we ran our custom Bro rule against the pre-captured dataset, we were able to successfully detect botnet communications that used either the DGS or FFSN technique with a false positive rate of < 5%. However, we have set the threshold quite conservatively, increasing which will reduce false positives in our experiment.

For the second stage of detection, we created Bro rules with signature matching. Our approach was to detect SSH handshake between bots and C&C servers tunneled within DNS request and reply. We deployed DNS2TCP, a tool designed to relay TCP connections using DNS tunnel. We set up DNS2TCP clients and server in a virtualized lab environment, generated SSH traffic, captured traffic using Wireshark, and used Base-64 decoder to decode the packets, and created a signature. Figure 2 shows the Wireshark packet capture with SSH signature.


DNS2TCP.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
Expression...

```
No.  Time     Source        Destination   Protoco Length Info
170 3.664458 172.31.36.28  52.52.65.253  DNS  95 Standard query 0x1997 TXT AAAAAF2sAA.=auth.tunnel2.ialabs.net
172 3.689240 52.52.65.253  172.31.36.28  DNS 141 Standard query response 0x1997 TXT AAAAAF2sAA.=auth.tunnel2.ialabs.net TXT
175 3.689531 172.31.36.28  52.52.65.253  DNS 148 Standard query 0x6b61 TXT l1GFgAABAEYxNDFCREQ0NjIwNTI5MDlDMDNDQUU2QzBDOTRFRTc2QjVDOUY1MDE.=auth.tunnel2.ialabs.net
178 3.713012 52.52.65.253  172.31.36.28  DNS 173 Standard query response 0x6b61 TXT l1GFgAABAEYxNDFCREQ0NjIwNTI5MDlDMDNDQUU2QzBDOTRFRTc2QjVDOUY1MDE.=auth.tunnel2.ialabs.net TXT
182 3.713248 172.31.36.28  52.52.65.253  DNS 102 Standard query 0x31be TXT l1ED4t+cAHNzaA.=connect.tunnel2.ialabs.net
184 3.736784 52.52.65.253  172.31.36.28  DNS 127 Standard query response 0x31be TXT l1ED4t+cAHNzaA.=connect.tunnel2.ialabs.net TXT
186 3.737225 172.31.36.28  52.52.65.253  DNS  89 Standard query 0x3afc TXT l1EAAAABBA.tunnel2.ialabs.net
187 3.737364 172.31.36.28  52.52.65.253  DNS 144 Standard query 0x959a TXT l1EAAAACCFNTSC0yLjAtT3BlblNTSF83LjJwMiBVYnVudHUtNHVidW50dTIuMg0.K.tunnel2.ialabs.net
188 3.737452 172.31.36.28  52.52.65.253  DNS  89 Standard query 0x79e1 TXT l1EAAAADBA.tunnel2.ialabs.net
192 3.760489 52.52.65.253  172.31.36.28  DNS 169 Standard query response 0x3afc TXT l1EAAAABBA.tunnel2.ialabs.net TXT
194 3.760710 172.31.36.28  52.52.65.253  DNS  89 Standard query 0xd0b2 TXT l1EAAQAEBA.tunnel2.ialabs.net
195 3.760950 52.52.65.253  172.31.36.28  DNS 419 Standard query response 0x959a TXT l1EAAAACCFNTSC0yLjAtT3BlblNTSF83LjJwMiBVYnVudHUtNHVidW50dTIuMg0.K.tunnel2.ialabs.net TXT
196 3.760995 52.52.65.253  172.31.36.28  DNS 472 Standard query response 0x79e1 TXT l1EAAAADBA.tunnel2.ialabs.net TXT
197 3.761232 172.31.36.28  52.52.65.253  DNS  89 Standard query 0xc742 TXT l1EAAgAFBA.tunnel2.ialabs.net
198 3.761683 172.31.36.28  52.52.65.253  DNS 253 Standard query 0x2f7e TXT l1EAAwAGCAAABTQFFN181y9Cf75mgJOszMTo/5UAAADEY3VydmUyNTUxOS1zaGE.yNTZAbGlic3NoLm9yZyxlY2RoLXNoYTItbmlzdHAyNTYsZWNkaC1zaG…
199 3.761759 172.31.36.28  52.52.65.253  DNS  89 Standard query 0x09ce TXT l1EAAAAHBA.tunnel2.ialabs.net
200 3.761840 172.31.36.28  52.52.65.253  DNS 253 Standard query 0x43f9 TXT l1EAAAAICGxtYW4tZ3JvdXAtZXhjaGFuZ2Utc2hhMjU2LGRpZmZpZS1oZWxsbWF.uLWdyb3VwLWV4Y2hhbmdlLXNoYTEsZGlmZmllLWhlbGxtYW4tZ3JvdX…

0000  [redacted]                                         .C\..... ^.t...E.
0010                                                     ..S.@.@. .,..$.44
0020                                  95 9a 01 00 00 01  A..^.5.n F.......
0030  00 00 00 00 00 00 3f 6c  6c 45 41 41 41 41 43 43   ......?l lEAAAACC
0040  46 4e 54 53 43 30 79 4c  6a 41 74 54 33 42 6c 62   FNTSC0yL jAtT3Blb
0050  6c 4e 54 53 46 38 33 4c  6a 4a 77 4d 69 42 56 59   lNTSF83L jJwMiBVY
0060  6e 56 75 64 48 55 74 4e  48 56 69 64 57 35 30 64   nVudHUtN HVidW50d
0070  54 49 75 4d 67 30 01 4b  07 74 75 6e 6e 65 6c 32   TIuMg0.K .tunnel2
0080  06 69 61 6c 61 62 73 03  6e 65 74 00 00 10 00 01   .ialabs. net.....
```

Figure 2: Wireshark packet capture for DNS2TCP traffic

By analyzing packet No. 187 in the capture data, the string *AAACCFNTSC0yLjAtT3BlblNTSF83LjJwMiBVYnVudHUtNHVidW50dTIuMg0* was used to establish an SSH tunnel. This string is equivalent to *SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.2*. To detect other versions of SSH-2.0 connections and/or operating systems, only the *CFNTSC0yLjAtT3BlblNTSF8* part is used to detect SSH connections tunneled in DNS packets. After decoding this Base-64 encoded string, it is equivalent to *SSH-2.0-OpenSSH_*. To detect only *OpenSSH_* string, the string *T3BlblNTSF8* is used. This signature was used to write a custom Bro script to detect botnet communication tunneled within DNS.

On testing the rule with pre-captured dataset, our technique was able to successfully detect DNS-tunneled botnet communication with < 2% false positive. Since we did not use all variations of SSH signature, our false negative rate was higher (~10%), which can be reduced by implementing all variations of SSH signature in the Bro rule.

5 Conclusion and Future Extension

In this paper we presented a multi-staged approach to detect botnet communication. The novel approach combines both anomaly detection technique and signature matching to detect botnets during their rallying stage and also during their communication with C&C servers. We developed custom Bro rules to implement the detection technique and tested the approach on the dataset collected from the Stratosphere Lab.

Experimental results demonstrated that our approach was able to detect DGA and FF-based botnets and also DNS-tunneled botnets using SSH signature. There are several opportunities to improve the approach through future work. The choice of threshold is the first challenge we had. Although we used published statistics by Brandhorst and Pras [14] on domain name service traffic, this threshold is not perfect and provides only preliminary results. The system should be able to dynamically detect these thresholds based on system behavior, which can be a future extension of this research. At the same time, different variations of SSH signature need to be implemented in the rulesets to improve false negatives for signature matching.

References

[1] Aditya K. Sood, Sherali Zeadally. A Taxonomy of Domain-Generation Algorithms. IEEE Security and Privacy. Vol. 14 (4), July-Aug 2016.

[2] Jose Nazario, Thorsten Holz. As the Net Churns: Fast-Flux Botnet Observations. In Proceedings of 3rd International Conference on Malicious and Unwanted Software. VI, USA. October 2008.

[3] Bro Network Security Monitor. https://www.bro.org/. Retrieved on November 2018.

[4] Vojtech Krmicek. Inspecting DNS Flow Traffic for Purposes of Botnet Detection. GEANT3 JRA2 T4 Internal Deliverable (2011): 1-9.

[5] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium, NDSS '11,* San Diego, California, 2011.

[6] Christian J. Dietrich, Christian Rossow, Felix C. Freiling, Herbert Bos, Maarten van Steen, and Norbert Pohlmann. On botnets that use DNS for command and control. In *Proceedings of the 2011 Seventh European Conference on Computer Network Defense* 57, pp. 9-16, Washington, DC, 2011.

[7] Sergii Lysenko, Oksana Pomorova, Oleg Savenko, Andrii Kryshchuk, and Kira Bobrovnikova. DNS-based anti-evasion technique for botnets detection. In *Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (pp. 453-458), Warsaw, Poland, 2015.

[8] Y. Jin, H. Ichise, & K. Iida. Design of detecting botnet communication by monitoring direct outbound DNS queries. In *Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)* (pp. 37-41), Washington, DC, 2015.

[9] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, & G. Eaton. Real-time detection of fast flux service networks. In *Proceedings of Cybersecurity Applications & Technology Conference for Homeland Security* (pp. 285-292). Washington, DC, 2009.

[10] M. Dabbagh. *Covert channels in botnets*. Retrieved from http://web.engr.oregonstate.edu/~dabbaghm/projects/CovertChannels.pdf. 2018.

[11] H. Zhang, C. Papadopoulos, & D. Massey. Detecting encrypted botnet traffic. In *Proceedings of IEEE INFOCOM*, 2013, doi:10.1109/infcom.2013.6567180, 2013.

[12] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, & D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA Based Malware. In *Proceedings of the 21st USENIX Conference on Security Symposium* (pp. 24-24). Bellevue, WA, 2012.

[13] S. Yadav, & A. L. Reddy. Winning with DNS failures: Strategies for faster botnet detection. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Security and Privacy in Communication Networks* (pp. 446-459). Doi:10.1007/978-3-642-31909-9_26, 2012.

[14] C. J. Brandhorst & A. Pras. DNS: A statistical analysis of name server traffic at local network-to-Internet connections. In C. Delgado Kloos, A. Marín, & D. Larrabeiti (Eds.), *EUNICE 2005-Proceedings of the 11th Open European Summer School and IFIP WG6.4/6.6/6.9 Workshop* (pp. 255-270), 2006.

[15] S. Garcia, S. *Malware capture facility project*. Retrieved August 2017 from https://stratosphereips.org.