Holistic Model for HTTP Botnet Detection based on DNS Traffic Analysis

Abdelraman Alenazi[1], Issa Traore[1], Karim Ganame[2], Isaac Woungang[3]

[1]University of Victoria, ECE Department, Victoria, BC Canada

aalenazi@uvic.ca, itraore@ece.uvic.ca

[2]StreamScan, 2300 Rue Sherbrooke E, Montreal, QC, Canada

kganame@streamscan.io

[3]Ryerson University, Dept. of Computer Science, Toronto, ON, Canada

iwoungan@scs.ryerson.ca

Abstract. HTTP botnets are currently the most popular form of botnets compared to IRC and P2P botnets. This is because, they are not only easier to implement, operate, and maintain, but they can easily evade the detection. Likewise, HTTP botnets flows can easily be buried in the huge volume of legitimate HTTP traffic occurring in many organizations, which makes the detection harder. In this paper, a new detection framework involving three detection models is proposed, which can run independently or in tandem. The first detector profiles the individual applications based on their interactions, and isolates accordingly the malicious ones. The second detector tracks the regularity in the timing of the bot DNS queries, and uses this as basis for detection. The third detector analyzes the characteristics of the domain names involved in the DNS, and identifies the algorithmically generated and fast flux domains, which are staples of typical HTTP botnets. Several machine learning classifiers are investigated for each of the detectors. Experimental evaluation using public datasets and datasets collected in our testbed yield very encouraging performance results.

Keywords: HTTP botnet, botnet detection, machine learning, passive DNS, DGA domains, malicious fast flux DNS.

1. Introduction

A botnet is a network of enslaved machines, distributed geographically, which may be directed to perform malicious actions against potential targets at a large scale [19]. The enslaved machines are compromised hosts known as bots. The individual or group of individuals controlling those machines are known as botmaster. Early botnets used centralized architecture for exchanging command and control (C&C) messages. The most prevalent communication protocols used in those

earlier botnets was the Internet Relay Chat (IRC). However, this type of botnet is easy to detect and disrupt due to the single point of failure embodied by the IRC server, which manages the C&C communications. Once the server is shut down, the botmaster loses control of the network of bots.

The next generation of botnets, which appeared a decade ago, addressed the aforementioned weakness by using peer-to-peer (P2P) protocols for command and control. Due to its distributed and resilient control structure, a P2P botnet is harder to shut down than an IRC-controlled botnet. However, in the recent years, as more knowledge has been acquired about P2P botnets, more effective solutions have been proposed to detect them and mitigate their impact. Furthermore, P2P botnets are more complex to implement, deploy and operate. As a result, there have been a shift in the C&C protocol of modern botnets from IRC and P2P channels to websites, using HTTP [6].

The advent of HTTP botnet can be linked to the development of exploit kits (EKs). EKs are sophisticated malicious software platforms, often professionally developed and marketed in the dark web, which allow cybercriminals to readily build and operate botnets and other types of malicious software. Due to the prevalence of http communications and sites, detecting botnets that use HTTP protocols is much harder [3, 4, 7, 8]. Many organizations host websites for regular business activities, and as such, enable http communications. Hence, it is easy for http-based botnets to evade the detection by hiding their command and control messages in the huge volume of legitimate HTTP traffic occurring in most organizations.

Despite such challenge, HTTP botnets have certain characteristics which can be leveraged and used to detect them. Some of those characteristics are rooted in the central role played by the concept of domain names in any web communication. The domain name by design represents a user-friendly mechanism to identify the servers in the cyberspace. However, user friendliness matters only when humans are involved in both ends. Such consideration does not matter in the automated settings embodied by botnets. As such, HTTP botnets abuse as much as possible the domain name system (DNS) toward achieving their main purpose of evading the detection, by using mechanisms such as fast flux DNS and algorithmically generated domain names. Ironically, while the DNS is abusively used by HTTP botnets to escape the detection, they leave a number of trails in the generated DNS traffic as part of the C&C communications, such as the regularity in the type and timing of the DNS requests, which are clues about their presence.

In this paper, we propose, to capture and analyze passive DNS queries for HTTP botnets detection [16]. We take a holistic approach by investigating different detectors which are suitable for capturing specific patterns. The combination of these detectors in a multi-detection scheme can provide a powerful HTTP botnet detection system, which can cover different types of HTTP botnets.

The rest of the paper is structured as follows. Section 2 summarizes some related work on HTTP botnet detection and the use of passive DNS for detecting the malicious activities. Section 3 provides some background on DNS and discusses some key characteristics of HTTP botnets that can be useful in building an ade-

quate detection scheme. Section 4 introduces our proposed holistic detection framework by describing the feature space involved in the different detectors. Section 5 presents our evaluation datasets and performance results. Section 6 concludes the paper and discusses some future work.

2. Related Works

2.1 On HTTP Botnets Detection

Tyagi et al. [13] proposed a technique, which detects HTTP botnets by tracking similar flows occurring at regular time intervals, from the C&C to the bots in response to HTTP GET requests. The similarity between two flows is determined by computing and comparing the corresponding byte frequencies, for given sequence of bytes of length *N*. Although, the authors claim to have proposed the first approach that can be applied to all types of HTTP botnets, it was tested using only the Zeus botnet, yielding a detection rate (DR) of 100%, and False Positive Rate (FPR) of 0% for traffic with static C&C IP addresses, and DR=98.6% and FPR=0% for traffic with dynamic C&C IP addresses (fast flux).

Haddadi et al. [8] extracted and analyzed some flow-based features of botnets using two different classifiers, C4.5 and Naive Bayes. The flows are extracted by aggregating the network traces using flow exporters and using only packets headers. The experimental was done by collecting the domain names, including regular domains from the Alexa site and the malicious botnet C&C domain names generated by two different HTTP botnets, namely, Zeus and Citadel. A dataset was generated by running a custom script to initiate the HTTP connections with the domain names from the compiled list. It is unclear, however, whether the corresponding traffic really captures the actual behavior of the corresponding botnets. The evaluation without/with HTTP filtering (i.e. removing non-HTTP related traffic) indicated that the latter yields improved performance. The best performance was obtained when using the C4.5 classifier with the HTTP filter, yielding a performance of (DR=97%, FPR=3%) for Citadel and (DR=86%, FPR=15%) for Zeus.

Cai and Zhou [3] proposed a model which starts by clustering the HTTP request data from the HTTP flows using the Levenshtein Distance metric. Their approach was validated by using a dataset that they collected by themselves, yielding a False Alarm Rates (FAR) ranging from 13.6% to 26.3%.

Khillari and Augustine [9] proposed an HTTP botnet detection technique by mining the patterns set from the network traffic using the Apriori algorithm. However, it is unclear how the proposed approach was validated experimentally.

Garasia et al. [7] also proposed an HTTP botnet detection approach by applying the Apriori algorithm to the network traffic generated from HTTP communications. Although an attempt was made to conduct some experimental validation, the focus was limited to a hypothetical example. No real malicious samples were involved in the study, and no performance results were provided.

Venkatesh and Anitha [15] studied the detection of HTTP botnet by applying an Adaptive Learning Rate Multilayer Feed-forward Neural Network to

relative and direct features of TCP connections. An experimental validation was based on a dataset consisting of botnet traces for Spyeye and Zeus, that was merged with normal web traffic collected separately. A performance of (DR=99%, FPR=1%) was obtained. Next, the use of C4.5 decision tree, Random Forest and Radial Basis Function, confirmed that the proposed neural network model showed promising results.

2.2 On DNS Traffic Monitoring

Piscitello [10] discussed about how DNS traffic monitoring can help uncover indicators of compromise (IOC) for malware such as Remote Access Trojans (RATs). Specifically, six signs and traffic patterns for suspicious activities were identified. One of these signs relates to DNS queries that request known malicious domains or names with characteristics common to Domain Generation Algorithms (DGA) associated with the botnet. Other signs are the abnormally high amount of query answers returning Non-Existent Domain (NXDOMAIN), the high amount of query responses with short time-to-live (TTL) for newly registered domain names, and responses with suspicious IP addresses.

Weymes [17] proposed an approach to detect suspicious DNS queries by considering three different features consisting of domain length, name character makeup and level domain (e.g. .com, .ru, .biz, etc.). Based on a traffic sample from the Zeus botnet, it was shown that the domain length can raise a red flag when it exceeds 12 characters long. According to Weymes, a typical Zeus query is more than 33 characters long whereas normal queries are less than 12 characters long. The name character makeup such as alphanumeric, numbers, or vowels only, was also considered as impactful.

No experimental models or performance results for malicious activities detection was provided for the above work on passive DNS, however, some signs and characteristics that can be taken into account when attempting to detect suspicious DNS queries were highlighted.

Da Luz [5] used passive DNS traffic to detect domain names related to botnet activities. In their proposed model, 36 different features are extracted from passive DNS data and processed using machine learning techniques. The extracted features consist of lexical features of the domain name (e.g. number of characters, number of digits, number of consonants, statistics of character n-grams, to name a few) and network features of the domain such as TTL and number of IP subnetworks. Three different techniques are studied, including the k-Nearest Neighbors (kNN), the decision trees and the Random Forests. Experimental evaluation conducted using a 2-week passive DNS traffic yielded an accuracy rate of 97%, and a False Positive Rate (FPR) of 3%.

Antonakakis et al. [1] proposed a dynamic reputation system for DNS called Notos, assuming that malicious DNS activities are distinguishable from le-

gitimate DNS traffic. Based on passive DNS data, the system extracts and analyzes the network and zone features of domains, then it constructs models of known legitimate domains and malicious domains. A domain is classified as legitimate or malicious according to a reputation score computed from the constructed models. Their proposed system was evaluated on a large ISP network with DNS traffic of 1.4 million users. Notos detects malicious domains with TPR=96.8% and FPR=0.38%. It was also reported that Notos can identify malicious domains weeks before they go active.

In [2], Bilge et al. developed a system called *Exposure* for detecting malicious domains, which uses 15 behavioural DNS attributes distributed among four groups, namely: Time-based (e.g. short-life, daily similarity), DNS answer-based (e.g. number of distinct resolved IP addresses, number of domains sharing the same IP), TTL-based (e.g. TTL, average and standard deviation, and number of TTL change), and DNS name-based features (e.g. measuring the percentage of numerical characters in a domain name). For the design of such system, the J48 decision tree was used for classification purpose assuming that short-lived domains[1] are often malicious. The proposed model was evaluated on an ISP dataset of 100 billion DNS packets, and yielded a performance of DR=98% with FPR=1%.

Nazario and Holz [11] proposed a way to classify fast flux domains using a series of heuristic rules. Their detection model extracts nine different features from DNS resources records such as TTL values, the discovery of more than five unique IPs in a single A-record query response, the average distance between IP addresses in A-records query responses, to name a few. Domain names are classified as fast flux when they match four or more of the nine features. From this study, it was reported that over 900 domain names that are using fast flux technology can be identified.

3. Passive DNS and HTTP Botnet

3.1 DNS Overview

The Domain Name System is a hierarchical scattered system that is mainly responsible for translating and mapping meaningful domain names to IP addresses. It is a critical component of the Internet infrastructure that is currently being used in most of the network services. Besides the ease of domain name memorization, DNS makes it possible to link a domain name to a collection of services such as Mail Exchange server, Web server, File servers or other Internet resources in a

[1] Short life refers the time interval between two queries of the same domain.

meaningful and independent way. The domain name system provides stability in how Internet resources are referred and located even if the underlying infrastructure changes.

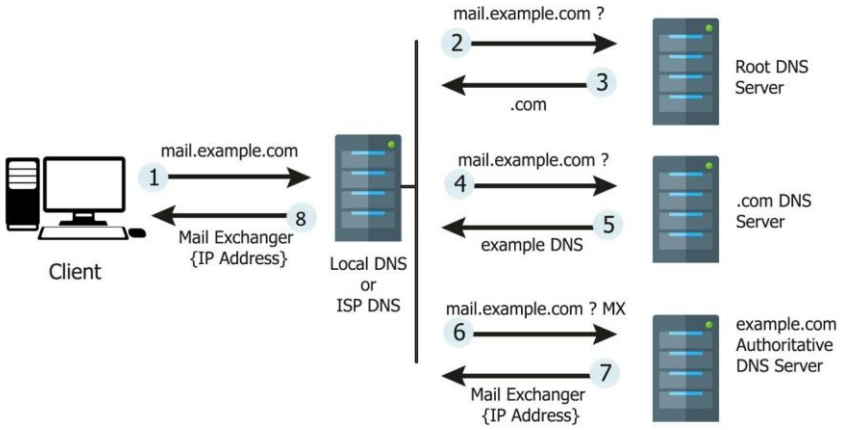


Figure 1. DNS Query Cycle

DNS communications are based on hierarchical recursive requests. When a user aims to establish a connection to a domain name (e.g. example.com), DNS client sends a query to a DNS recursive resolver, which may be hosted locally or by third-parties such as Internet Service Providers (ISPs). DNS recursive resolvers attempt initially to resolve the received queries using cached information from past queries. If such resolution is unsuccessful, the request will be forwarded to other servers iteratively until a match is found. Figure 1 shows the process of DNS resolving when no cached records are available.

DNS naming structure is shaped as tree data structure. A top-level-domain (TLD) is the node that comes after the root. For example, .com, .net, and so on, are known as TLDs. Each TLD is a registry that holds and manages a zone file. A prefix name or sub domain of each TLD is known as a second level domain (SLD) name. All second level domains are controlled by authoritative DNS servers. A domain name can have one or more hierarchical sub domains; each sub domain level is defined by the incremental second-level domain. For example, foo.example.com is a third level domain. Moreover, a complete domain name (e.g. www.google.com or blog.example.com) is referred to as a fully qualified domain name (FQDN).

DNS provides different types of records which map various resources such as web, mail servers, etc. Each DNS query contains a time-to-live (TTL) val-

ue that determines how long a machine caches a query. Normal TTL values, for an A record are between 3,600 to 86,400 seconds.
Some web services use a technique known as fast flux DNS which sets TTL to lower values for the purpose of balancing the load between multiple servers. Fast flux DNS solves issues such as single point of failure, by distributing ingress traffic to multiple cloned servers or mapping several IP addresses to one domain name.

Cybercriminals use the same concept in order to evade the IP address blacklisting and achieve high-availability [19]. While fast flux techniques are used for legitimate purpose, cybercriminals have flipped this technology over its head, as they have found that a malicious usage of Fast Flux networks provides a very effective mechanism for hiding their C&C servers and ensuring resilience. This is done by using a short time-to-live TTL on DNS resources records, which allows swapping and rotating between servers efficiently. A key challenge faced by researchers is about developing effective approaches to differentiate between malicious and benign fast flux networks.

3.2 HTTP Botnet Architecture

Traditional botnet architectures such as IRC botnet and P2P botnets use push-style communications to exchange commands. The bots join the identified command and control channels to receive the commands from the botmaster, then remain connected to these channels. Such channels can be IRC servers for IRC botnet or other peers for P2P botnets. In contrast, HTTP botnets use pull-style communications to obtain the commands. Instead of being connected permanently to the channels, the bots regularly contact the HTTP C&C servers hosted on different sites in order to get the commands. The commands are embedded in web pages hosted on the C&C servers, and can be retrieved by the bots after requesting corresponding pages. While the HTTP bot does not remain connected to the C&C servers, it visits the corresponding sites on a regular basis, and at a pace defined by the botmaster. Such regularity can be leveraged in detecting the presence of the botnet.

Furthermore, HTTP botnet servers involve fewer web services compared to legitimate web servers. Typical HTTP botnet C&C server will provide a command download only while legitimate sites will support a wide range of services. The request parameters in HTTP C&C tend to be relatively stable or similar (e.g. images used as command files) while a variety of resources will be requested and exchanged in legitimate web communications.

4. Proposed Detection Model

4.1 General Approach

Our proposed detection model leverages the following key characteristics of HTTP botnets:

1. The reliance on pull-style communication model means that the bots initiate connections with the C&C server to get commands and updates. This will require issuing DNS queries, which could be tracked toward detecting the bots.
2. The regularity in the connection timing and characteristics. As connections to the C&C sites typically take place at regular time interval, analyzing the underlying timing information can help detect the presence of the bots.
3. The C&C domain names, selected only for the sole purpose of escaping detection, exhibit characteristics which are glaring departure from legitimate uses of domain names. Legitimate domains tend to be stable and user-friendly, whereas C&C domains are short lived and not intended for human consumption.
4. The extremely limited pool of requested web services by HTTP bots means predictable interaction patterns, which are different from what could be expected from legitimate applications' interactions. Such distinct interaction patterns can be captured in separate application profiles, and used to isolate and detect eventually botnets interactions.

Based on the aforementioned considerations, our approach involves multiple detectors, each capturing specific characteristics of the HTTP botnets. We propose specifically 3 different detectors as follows:

1. Time-series detector: its role is to detect the HTTP botnets by analyzing their timing characteristics.
2. Application detector: its role is to profile the individual applications running on the host based on their DNS interactions. This profile is used as a model to identify suspicious applications which eventually may be flagged as HTTP botnets.
   Domain Mass detector: its role is to analyze the characteristics of the domain names and identify the malicious domains that potentially could be part of the HTTP botnets.

These detectors work independently by extracting separate features from the data and classifying those features using a machine learning classification. We have investigated separately three different classifiers for all detectors, namely: the Random Forests, the J48 Decision Tree, and the Naïve Bayes.

Each of these detectors capture and analyze the packets flows over separate time windows. In our system design, we use time windows of 24 hrs, 1 min,

and 10 min for the application detector, the domain mass detector, and the time-series detector, respectively.

In the sequel, we describe the features space for each of the aforementioned detectors.

4.2 Domain Mass Detector

The domain mass detector analyzes the characteristics of the domain name involved in the DNS queries and attempts to identify malicious domains such as algorithmically generated domains or domains associated with malicious fast flux DNS.

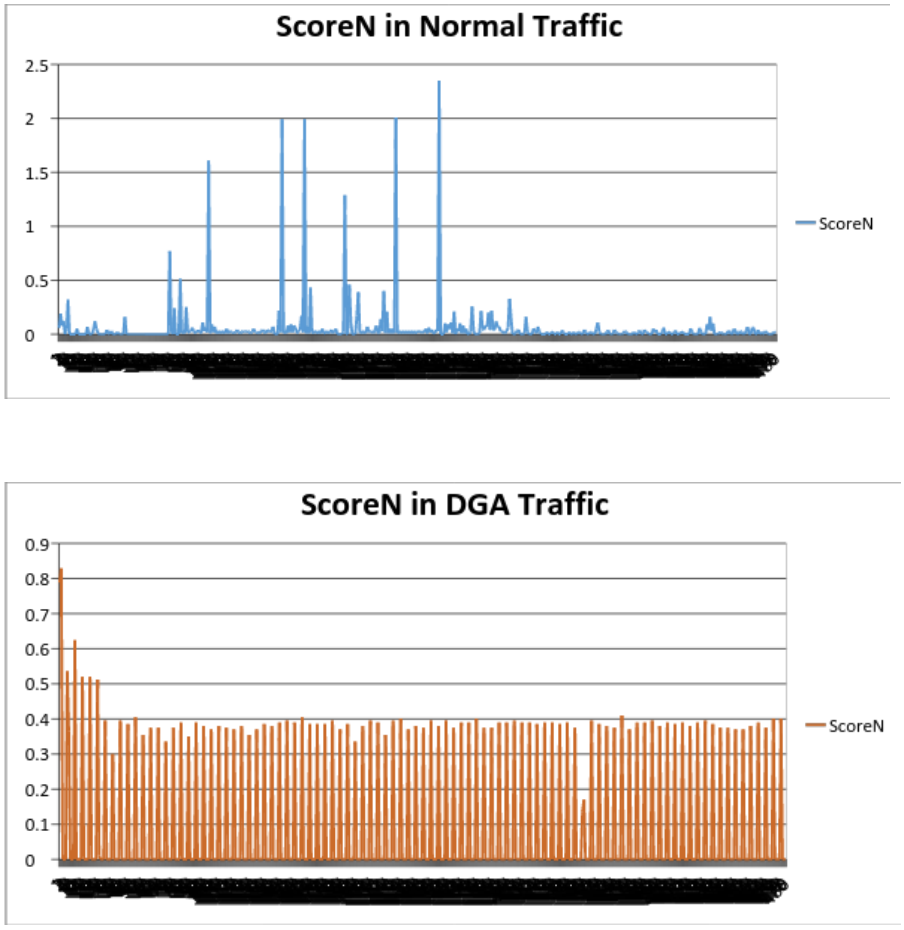


Figure 2. Normalized ratio (ScoreN) of nonexistent domain over successful queries for normal and DGA traffic samples

The underlying detection model extracts the following features:

- Total number of queries
- FQDN length (average, standard deviation, variance)
- Number of distinct DNS servers
- Number of geolocations of resolved IP addresses
- Number of A and AAAA records

One of the characteristics which stand out when comparing normal and malicious botnet traffic is the length of the involved domain names. As mentioned earlier, due to the use of DGAs, such domain names disregard basic DNS concepts such as user friendliness.

Normal DNS queries have varying domain lengths, which according to Weymes [17], are typically below 12 characters long. In contrast, botnets queries involving DGAs are often much longer, with limited variation. Therefore, we include in the feature set for the domain detector the *average domain length*, and the corresponding *standard deviation* and *variance* over a packet flow.

A recurring characteristic of a botnet architecture involving DGA is the relatively high number of non-existent internet domain name since the bot generates fake queries in order to evade detection. These fake queries trigger a high number of NXDOMAIN results (i.e. nonexistent domain names). If we consider the number of NXDOMAIN as a feature, it might work well on small networks, however, it may be lacking on larger networks.

Let *ScoreN* be the normalized ratio between the number of NXDOMAIN by the number of NOERROR (i.e. successful queries) observed over a time window, i.e.

$$ScoreN = \frac{(\text{Number of NXDOMAIN})}{(\text{Number of NOERROR}) \times 100}$$

Figure 2 shows the normalized score applied on the normal and malicious traffic samples, respectively. It is observed that there are some clear differences between malicious and non-malicious traffic based on the *ScoreN* metric. Furthermore, it can be noted that the number of DNS queries in the normal traffic is very low compared to the HTTP botnet traffic, in particular, the ones using DGA and/or malicious fast flux DNS, which is the case for virtually all the existing ones. Indeed, the infected machines do constantly notify their C&C servers in order to update their active status. Also, in leveraging the fast flux techniques, the C&C servers are assigned to domain names with low TTL by the botmaster, which result in constant DNS lookups. Therefore, it can be concluded that the *total number of queries* is a useful indicator of compromise.

The type of queries is also an interesting characteristic. In fact, normal activities have a variety of DNS query types while botnet traffic usually involves the same type of records. Therefore, counting the *total number of A and AAAA* records can be a useful indicator of compromise as well.

Another interesting characteristic is the number of distinct DNS servers found in the queries. Some bot-infected machines attempt to query different DNS servers. Normally, all DNS queries are redirected to the default DNS server set by the network administrator or by the host. Queries which are redirected to different DNS servers may suggest the presence of a malicious behavior.

Legitimate fast flux domains are typically co-located around the same geographic areas, for instance, they can be associated with the data centers operated by organizations or service providers using such technology. In contrast, malicious fast flux domains are scattered around the world. Those domains tend to be distributed geographically depending on the locations of the compromised machines used as proxies or used to host the botnet C&C servers. This aspect is captured in our model by the number *of geolocations of the resolved IP addresses.*

4.3 Application Detector

The application detector profiles the individual applications deployed on a host that can potentially be the target of bot infection. The detector tracks the DNS requests of legitimate software applications that require retrieving updates from remote servers, then detects the misbehaving application profiles.

We have studied the DNS interactions for individual applications when the host machine is in an idle state, i.e. when the machine is running and is not being used by a human user (e.g. to browse on the web). By studying the DNS traffic behavior, we have identified a number of features which allow profiling the individual applications. The detector maintains a database of known/legitimate applications' profiles, then flag as potentially malicious those applications which do not fit the profiles. By checking the communication protocols used by an identified suspicious application, we have determined whether it is a known or new form of HTTP botnet.

The detector computes the following features:

- FQDN length
- Domain level
- Query type
- TTL value
- Repeat query count

The *repeat query count*, one of the features in our model, is the number of times a DNS query (i.e. request with same characteristics) has been queried within the cached period (i.e. TTL – the time it was set to be cached).

7:59:34	192.168.50.14	8.8.8.8	IN	armmf.adobe.com.	CNAME	ssl.adobe.com.edgekey.net.	278
7:59:34	192.168.50.14	8.8.8.8	IN	ssl.adobe.com.edgekey.net.	CNAME	e4578.b.akamaiedge.net.	19922
10:05:00	192.168.50.14	8.8.8.8	IN	ardownload.adobe.com.	CNAME	ardownload.wip4.adobe.com.	10587
10:05:00	192.168.50.14	8.8.8.8	IN	ardownload.wip4.adobe.com.	CNAME	ardownload.adobe.com.edgesui	10
10:04:51	192.168.50.14	8.8.8.8	IN	ssl.adobe.com.edgekey.net.	CNAME	e4578.b.akamaiedge.net.	21460
10:04:51	192.168.50.14	8.8.8.8	IN	armmf.adobe.com.	CNAME	ssl.adobe.com.edgekey.net.	293

(a) Sample of Skype CNAME Queries

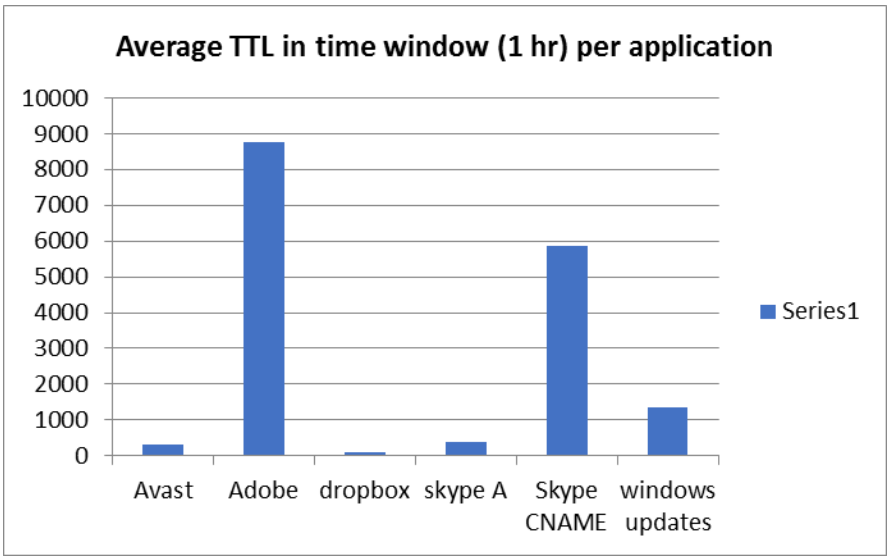
7:39:30 AM	192.168.50.11	8.8.8.8	IN	auth.ff.avast.com.	A	77.234.41.21	299
7:40:43 AM	192.168.50.11	8.8.8.8	IN	vl.ff.avast.com.	A	77.234.41.59	299
7:40:43 AM	192.168.50.11	8.8.8.8	IN	vl.ff.avast.com.	A	77.234.41.53	299
7:40:43 AM	192.168.50.11	8.8.8.8	IN	vl.ff.avast.com.	A	77.234.41.74	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.35	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.26	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.23	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.34	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.25	299
7:50:00 AM	192.168.50.11	8.8.8.8	IN	su.ff.avast.com.	A	77.234.41.24	299
9:41:17 AM	192.168.50.11	8.8.8.8	IN	ipm-provider.ff.avast.com.	A	77.234.41.88	299
9:41:17 AM	192.168.50.11	8.8.8.8	IN	ipm-provider.ff.avast.com.	A	77.234.41.92	299
9:41:17 AM	192.168.50.11	8.8.8.8	IN	ipm-provider.ff.avast.com.	A	77.234.41.93	299
11:01:23 AM	192.168.50.11	8.8.8.8	IN	vl.ff.avast.com.	A	77.234.41.55	299
11:01:23 AM	192.168.50.11	8.8.8.8	IN	vl.ff.avast.com.	A	77.234.41.56	299

(b) Sample of Avast A Queries

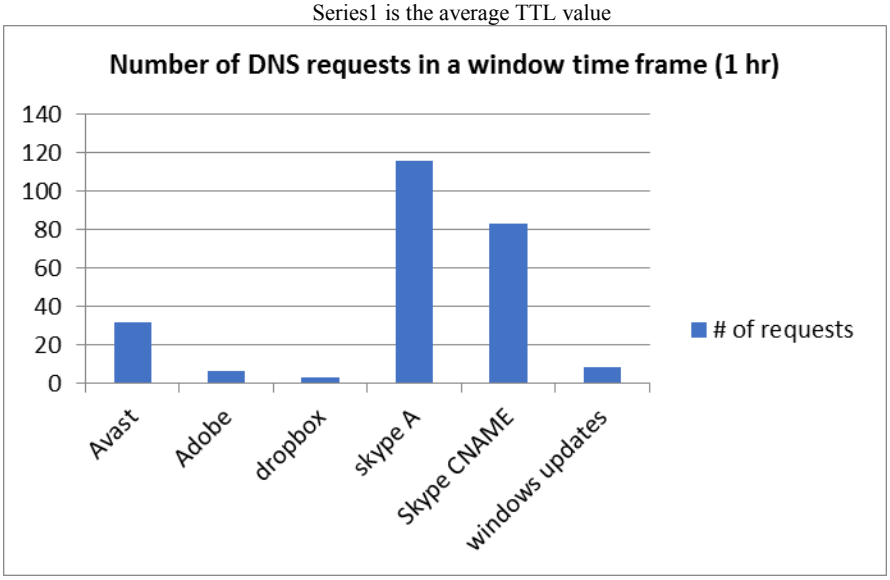
Figure 3. Sample of DNS queries for different applications

Figure 3 shows the sample DNS queries for different applications. It can be observed that specific query pattern is issued at certain times for each application. The query pattern for Skype contains only CNAME queries whereas that for Avast involves only A requests. Likewise, we have used the *query type* as a feature in profiling the individual applications.

The TTL value appears to be a distinct characteristic of each application. By analyzing the sample data, we have noticed that the FQDNs of each application have almost the same TTL cache as shown in Figure 3 and Figure 4. For instance, all DNS queries of Avast shown in Figure 3 have a TTL of 299 seconds, which may be due to the fact that these FQDNs are hosted on the same zone files. The TTL varies from one application to another, but each application has an almost stable number with a very small variation. Including the TTL value to our feature set helps characterizing such behavior.



Series1 is the average TTL value



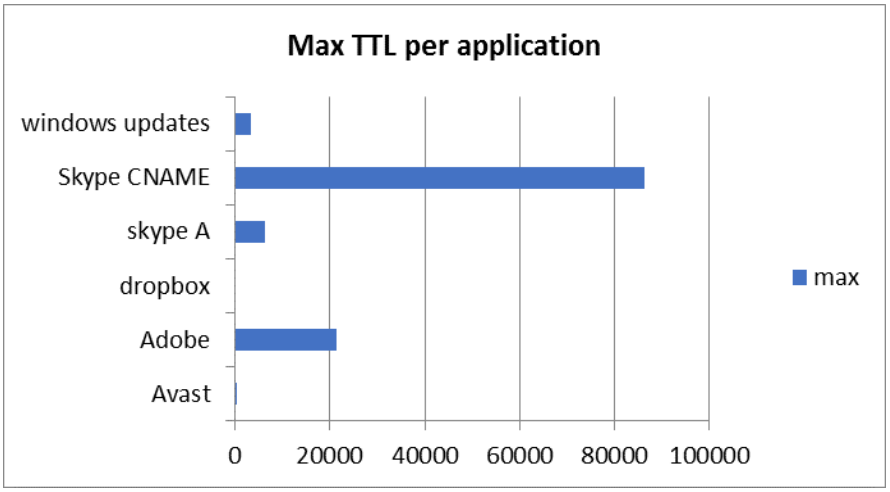


Figure 4. Number of queries and TTL statistics for different applications over sample traffic

The *domain level* considered in our feature model is an interesting discriminator between malicious domains and normal domains. As shown in Figure 3, normal application queries are often structured as third-level domain, fourth-level domain or even more. In contrast, the FQDN queries are usually second-level domains (SLDs). For example, Table 1 shows the C&C servers using the second level domain.

Table 1. Examples of C&C domain names found in HTTP botnet traffic sample

Domain name	Botnet Name
mxxywhxoc.cc	Papras Trojan
dyybbsux.cc	Papras Trojan
jmexlakjdk.cc	Papras Trojan
kuhfkadnmaxr.cc	Papras Trojan
saxtostfsa.cc	Papras Trojan
qudjmojvow.cc	Papras Trojan
tqqpteoxlcih.cc	Papras Trojan
cmpzygrl.cc	Papras Trojan

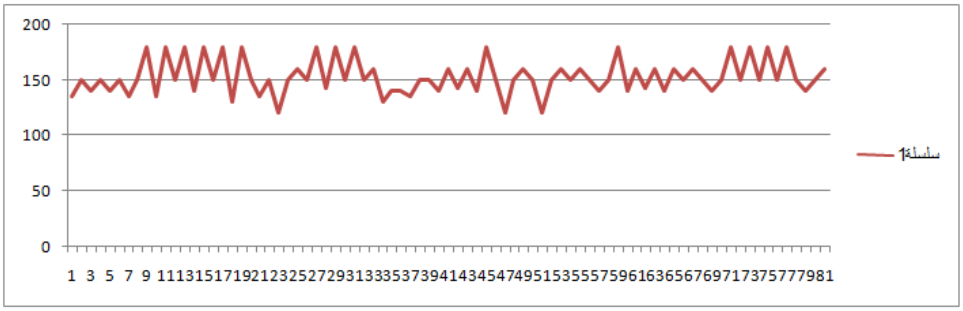
4.4 Time Series Detector

As mentioned earlier, there is some regularity in the timing behavior of the HTTP bots. This is a side effect of the underlying architecture which uses pull-style communications. By analyzing the HTTP botnet traffic sample, we have noticed that almost every bot communicates with its C&C server on a scheduled time. As

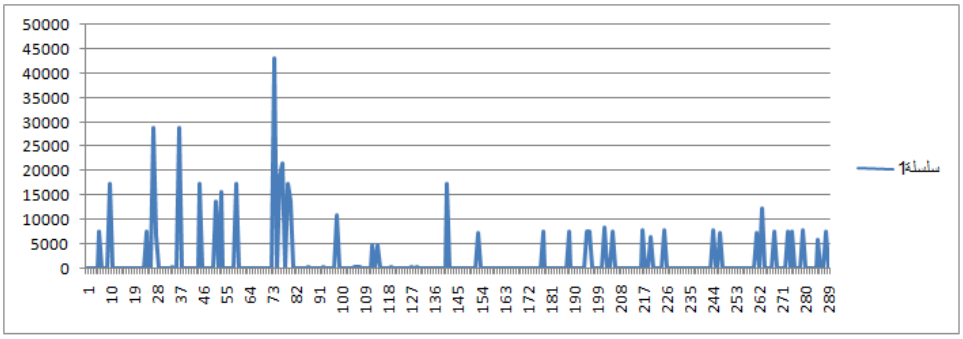
a result of this action, we have developed a detector that analyzes the DNS traffic flow based on short time interval between two queries. In this period of time, each query can be tracked and different statistical features can be extracted as follows:

- Total number of queries
- Time interval between successive queries (average, standard deviation, and variance)

Figure 5 depicts the average time interval computed over the normal and malicious traffic samples. It can be observed that there is a clear difference in the traffic patterns.



Graph showing average interval time in seconds of botnet DNS queries



Graph showing average interval time in seconds of Normal DNS queries

Figure 5. Average time interval over normal traffic vs. malicious traffic samples

5. Experimental Evaluation

5.1 Datasets

Since our proposed models can detect and classify different types of DNS activities, we have collected data from different public sources. However, not all the features considered in our models can be tested using the available public datasets. Therefore, we collected complementary datasets in our lab in order to achieve a full coverage of all the implemented detectors.

5.1.1 Public Datasets

We have acquired malicious traffic data from the Stratosphere IPS Project (https://stratosphereips.org/). This dataset was generated from a 8-days of operation of the Papras Trojan traffic that was running on a Windows XP machine. It contains a number of C&C calls and hard-coded DNS server (instead default DNS). It also contains only malicious traffic. Therefore, we have collected background normal traffic from our lab over three days and merged it with the malicious traffic from the Stratosphere IPS Project.

The size of the malicious dataset was 2,088,916 packets whereas that of the normal traffic was 30,853 packets, for a total of 2,119,769 packets. The combined dataset was used to evaluate the domain detector introduced earlier.

5.1.2 Complementary Datasets

In order to collect complementary datasets of real malicious and normal DNS traffic, we have designed a virtual environment in our lab, as shown in Figure 6, then used it to deploy different HTTP botnet exploit kits and legitimate software applications. The Exploit kits provide easy-to-use builders that generate malicious software. We have deployed 9 different C&C servers and generated 9 different bots to communicate with their C&Cs. We have also deployed the following 9 HTTP botnet kits: *Zyklon, Blackout, Bluebot, Betabot, Duox, BlackEnergy, Citadel, Zeus*, and *Liyphera*.

The exploit kits include a bot builder package, which allows us to set certain settings such as the C&C host servers. To collect DNS requests from the deployed exploit kits, the domain name for each botnet was required. Using *bind9*, we have setup an Authoritative-Only DNS server for a domain name *botnet.isot*. The domain is operated locally, and we have assigned each C&C server to a sub domain of *botnet.isot*. Our purpose was to associate the infected machine (e.g. bots) to the domain names so that we can monitor the behavior of outgoing DNS lookups.

For instance, DNS A query of *zeus.botnet.isot* points to 192.168.50.101. For each of the deployed bots, we have assigned the domain names of their C&C servers, not their IP Addresses; otherwise no DNS traffic will be observed. Thus, our design will generate DNS queries according to the exploit kits networking design.

In order to test our proposed *application detector*, we have deployed 20 virtual machines, where each machine was running a specific application. Applications used during our data collection include Anti-Virus applications (Malware-

Byte, ByteFence, AVG), online chatting and instant messaging applications (Skype, Facebook Messenger), Internet Browsing Applications (Firefox, Chrome), and e-mail client applications (eM Client, Thunderbird), and other applications such as Adobe reader, Photoshop, and Dropbox. While each application was running, we captured their DNS traffic only. Among all the 20 applications that were deployed, only 16 of them generated the DNS traffic. These machines were deployed for 7 days. After collection, we eliminated background traffic such as Microsoft updates. We used a tool called PassiveDNS to parse the collected pcap files passively, and then used our own python-based tool to normalize and label the data according to the corresponding application. The collected applications data consisted of 56,190 DNS packets.

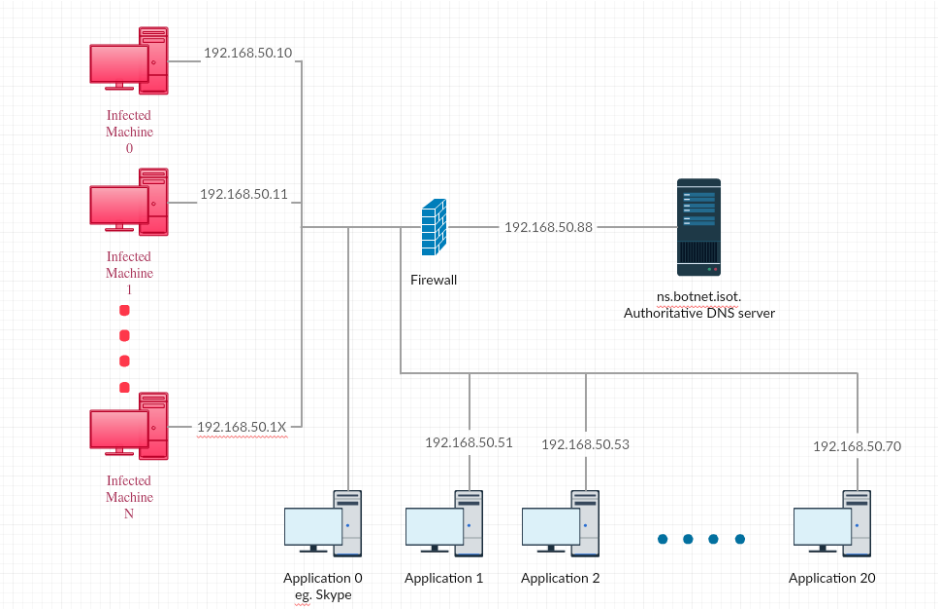


Figure 6. Data Collection Testbed

In order to evaluate the time series detector, we have deployed 9 different botnets to generate real botnet DNS traffic. The botnets ran for 5 days, and generated a total of 264,005 malicious DNS records. We also generated real normal DNS traffic in our experiment. The normal DNS capture ran for 3 days from one single machine while the botnet traffic came from 9 different infected machines. The total number of normal DNS traffic was 18,476 DNS records, yielding a dataset of 282,481 DNS packets in total.

5.2 Evaluation Results

As mentioned earlier, we studied 3 different classifiers, namely, Random Forests, J48 decision tree, and Naïve Bayes. We used an open source machine learning library for the Python programming language, which is called Scikit-learn. Each of the detectors ran using each of the aforementioned classifiers separately against the datasets described earlier. We ran each algorithm using a 10-fold cross validation, where in each run, the algorithm was trained on 90% of the data and 10% of the data was left for testing purpose. The results obtained are described in the sequel.

5.2.1 Domain Detector Evaluation

Table 2 shows the performance obtained for the domain detector. Random Forest shows promising results of 99.3% accuracy with 0.2% false positive rate.

The Gaussian Naïve Bayes (NB) achieved a lower performance compared to that of other classifiers. For instance, it has achieved a detection rate lower than that of the Random Forests by 4%. The Decision tree also showed a relatively better performance compared to the Gaussian NB, but not as good as that of the Random Forests.

Table 2. Domain Detector Evaluation Results

Algorithm	Detection Rate (%)	False Positive Rate (%)	Accuracy (%)
Gaussian Naïve Bayes	94.1	0.0	95.8
Random Forest	99.2	0.2	99.3
Decision Tree	99.2	0.3	99.3

5.2.2 Application Detector Evaluation

In the evaluation of the application detector, the traffic from known applications is flagged as normal whereas the traffic from unknown applications is considered malicious. In addition to the custom dataset collected in our lab which consists of HTTP botnets and known applications, we have also used the public malicious DGA traffic from stratosphereips.org.

It should be noted that the web browsing DNS traffic were excluded in our analysis because browsing traffic are generated from browsing, not from an active running application. Our detection is limited to any application running on the host machine, except web browsing. We did not considered the browsing data because such data is generated by human activities (non-automated). We make a distinction between data generated by a web browser application on its own such as getting updates (which is considered in our dataset) from data generated by a

user browsing the web using such application. Table 3 summarizes the performance results that were obtained.

Table 3. Application Detector Evaluation Results

Algorithm	Detection Rate (%)	False Positive Rate (%)	Accuracy (%)
Gaussian Naïve Bayes	6.18	2.44	50.35
Random Forests	94.85	3.90	95.44
Decision Tree	95.50	6.37	94.58

The obtained results consist of a detection rate of 94% and a false positive rate of 6.37% when using the decision tree algorithm. The random forests achieved an even better performance, yielding a detection rate of 95% and a FPR of 3.90%.

5.2.3 Time Series Detector

Table 4 depicts the evaluation results obtained for the time series detector. The Decision tree and Random Forests show promising results on average, achieving over 99% accuracy with very low false positive rates at 0.8% and 0.4%, respectively. The Gaussian Naïve Bayes performed relatively poorly.

Table 4. Application Detector Evaluation Results

Algorithm	Accuracy (%)	Detection Rate (%)	False Positive Rate (%)
Random Forests	99.5	99.5	0.4
Decision Tree	99.3	99.5	0.8
Gaussian Naïve Bayes	68.1	99.9	85.9

6. Conclusion

The HTTP botnet is much easier to develop and deploy compared to P2P and IRC botnets. Furthermore, due to the prevalence of HTTP communications, HTTP bots can evade the detection by using common HTTP ports, which are typically unfiltered by most firewalls. As a result, the detection of this type of botnet can be very challenging. In this paper, three different detection models have been designed, which perform relatively well on their own in detecting various types of HTTP botnets. However, each of these detectors operate using different time windows, ranging from short to longer time windows. Therefore, they can be considered as complementary detectors instead of competing ones. We have studied three different classifiers for each of the proposed detectors. Experimental results have shown

that the Random Forests yielded the best performance, with the Decision tree coming as a close second. The Gaussian Naïve Bayes performed poorly.

As future work, we plan to investigate how the proposed detectors can be integrated effectively in a multi-detector framework so that the detection accuracy is maximized. We also plan to investigate how these detectors operate when confronted with new forms of HTTP botnets unseen at the training time.

References

1. Antonakakis M., Perdisci R., Dagon D., Lee W., and N. Feamster." Building a Dynamic Reputation System for DNS". In the Proceedings of 19th USENIX Security Symposium (USENIX Security ’10), 2010.

2. Bilge L., Sen S., Balzarotti D., Kirda E., and C. Kruegel C., "Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains", ACM Transactions on Information and System Security (TISSEC), Volume 16 Issue 4, April 2014.

3. Cai T. and F. Zou, “Detecting HTTP botnet with clustering network traffic,” in Proc. 8th Conf. Wireless Commun., Netw. Mobile Comput., Sep., 2012, pp. 1–7.

4. Chaware S.P. and S. Bhingarkar, “A Survey of HTTP Botnet Detection”, International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 01, pages 713-714, Jan-2016.

5. da Luz P.M., “Botnet Detection Using Passive DNS”, Master Thesis, Department of Computing Science Radboud University Nijmegen, 2013/2014.

6. Fedynyshyn G., Chuah M.C., Tan G. (2011) Detection and Classification of Different Botnet C&C Channels. In: Calero J.M.A., Yang L.T., Mármol F.G., García Villalba L.J., Li A.X., Wang Y. (eds) Autonomic and Trusted Computing. ATC 2011. Lecture Notes in Computer Science, vol 6906. Springer, Berlin, Heidelberg

7. Garasia S.S., Rana D.P., and R.G. Mehta, “HTTP Botnet Detection using Frequent Pattern Set Mining”, International Journal of Engineering Science and Advanced Technologies, Volume-2, Issue-3, pages 619 – 624, May 2012.

8. Haddadi F., Morgan J., Filho E.G., and A. N. Zincir-Heywood “Botnet Behaviour Analysis using IP Flows with HTTP filters using classifiers”, 2014 28th International Conference on Advanced Information Networking and Applications Workshops, pages 7-12.

9. Khillari A. and A. Augustine, “HTTP-based Botnet Detection Technique using Apriori Algorithm with Actual Time Duration”, International Journal of

Computer Engineering and Applications, Volume XI, Issue III, March 17, pages 13-18.

10. Piscitello D., "Monitor DNS Traffic & You Just Might Catch A RAT", Dark Reading, UBM Technology, 6/12/2014, Web. June 27,2017, http://www.darkreading.com/attacks-breaches/monitor-dns-traffic-and-you-just-might-catch-a-rat/a/d-id/1269593

11. Nazario J. and T. Holz, "As the net churns: Fast-flux botnet observations", 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008. IEEE, 7-8 Oct. 2008. DOI: 10.1109/MALWARE.2008.4690854

12. Sood A.K., Zeadally S., and R.J. Enbody, "An Empirical Study of HTTP-based Financial Botnets", IEEE Transactions on Dependable and Secure Computing, vol. 13, pp. 236-251, March-April 2016, doi:10.1109/TDSC.2014.2382590

13. Tyagi R., Paul T., Manoj B.S., and B. Thanudas, "A Novel HTTP Botnet Traffic Detection Method", IEEE INDICON 2015

14. Tyagi A.K. and S. Nayeem, "Detecting HTTP Botnet using Artificial Immune System (AIS)", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA Volume 2– No.6, May 2012

15. Venkatesh G.K. and R. Anitha, "HTTP Botnet Detection using Adaptive Learning Rate Multilayer Feed-forward Neural Network", I. Askoxylakis, H.C. Pöhls, and J. Posegga (Eds.): WISTP 2012, LNCS 7322, pp. 38–48, 2012, International Federation for Information Processing (IFIP).

16. Weimer F., "Passive DNS Replication", In Proceedings of 1st Conference on Computer Security Incident, Singapore, 2005.

17. Weymes B., "DNS anomaly detection: Defend against sophisticated mal-ware", May 28, 2013; Web. June 28, 2017.https://www.helpnetsecurity.com/2013/05/28/dns-anomaly-detection-defend-against-sophisticated-malware/

18. Zhao D., Traore I., Sayed B., Lu W., Saad S., Ghorbani A., and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals", Elsevier Journal of Computers and Security, Volume 39, November, 2013, pages 2-16.

19. Zhao D. and I. Traore, "P2P Botnet Detection through Malicious Fast Flux Network Identification", 7th International Conference on P2P, Parallel, Grid,

Cloud, and Internet Computing -3PGCIC 2012, November 12-14, 2012, Victoria, BC, Canada