ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20161012115204
Original scientific paper

Fast-Flux Botnet Detection Based on Traffic Response and Search Engines Credit Worthiness

Davor CAFUTA, Vlado SRUK, Ivica DODIG

Abstract: Botnets are considered as the primary threats on the Internet and there have been many research efforts to detect and mitigate them. Today, Botnet uses a DNS technique fast-flux to hide malware sites behind a constantly changing network of compromised hosts. This technique is similar to trustworthy Round Robin DNS technique and Content Delivery Network (CDN). In order to distinguish the normal network traffic from Botnets different techniques are developed with more or less success. The aim of this paper is to improve Botnet detection using an Intrusion Detection System (IDS) or router. A novel classification method for online Botnet detection based on DNS traffic features that distinguish Botnet from CDN based traffic is presented. Botnet features are classified according to the possibility of usage and implementation in an embedded system. Traffic response is analysed as a strong candidate for online detection. Its disadvantage lies in specific areas where CDN acts as a Botnet. A new feature based on search engine hits is proposed to improve the false positive detection. The experimental evaluations show that proposed classification could significantly improve Botnet detection. A procedure is suggested to implement such a system as a part of IDS.

Keywords: Botnet; fast-flux; IDS

1 INTRODUCTION

The Domain Name System (DNS) is a hierarchical naming system for computers, services, clients, or any resource connected to the Internet. It provides a critical Internet service of mapping between two principal name spaces on the Internet: DNS tree and Internet protocol logical address space. DNS system is one of the unavoidable components of the critical infrastructure of the Internet as it makes possible to assign human-friendly names into logical Internet protocol addresses [1]. Maintaining consistency in handling names is one of the most important tasks in increasingly frequent changes of IP addresses due to widespread usage of dynamic DNS techniques.

Resolution, caching system and tree like organization provide the fault-tolerance ability for the DNS [2]. This fault-tolerance can help specific services to achieve availability by load distribution. The most often used example for this type of service is a web service. In this case, redundancy can be achieved if multiple web servers with the same web content are made available. This solution is named Round Robin DNS (RRDNS) [3]. In the case of the Round Robin algorithm for every DNS query a client is redirected to a different web server. - Time to live on authoritative DNS server decides when the redirection address for the web server will be changed. The main advantage of RRNDS is load redistribution between multiple servers.

To provide faster response time to the client by shortening network path from client to the server an Authoritative DNS server exploits Content Delivery Network (CDN) concept [4, 5]. In CDN concept every website is served on multiple web servers positioned around the globe. Using the previously established data about network connectivity upon client query for address, system responds using the server that has the smallest Round Trip Time [6]. Decision can be further improved by monitoring multiple servers. For instance, by examining server load or possible active threats. In case of Denial of Service attack (DoS) on one of the servers, others can provide content. Due to scattered network locations DoS attack against the web site becomes very difficult as it must be effective on every server in the field [7, 8].

This solution, as it is extremely usable for legitimate sites, it can also be exploited to compromised sites. A compromised computer, called bot, lies in the compromised network under control of a Botmaster [9]. Botmaster is assigning tasks over the network to the compromised computer or even mobile device. The communication between bots and Botmaster server plays a vital role in maintaining a Botnet's existence. Botnet is useless without the communication channel as they cannot join the bots' army and receive instructions and timing for the next goal of the Botmaster [10]. Obstructing the communication channel between bot and Botmaster can lead to rapid decrease of bot population and can result in failed malicious attack. In case of Denial of Service attack number of active bots guarantees a successful attack [11-12].

Botnets are continuously widening the approach in finding new infiltration communication channels. In the past, most popular channel of the attack was the Internet Relay Chat (IRC) protocol, which enabled a two-way communication between the bot and the Botmaster. To deliver the mission parameters only one-way communication is sufficient. An example of the Botmaster direction can be a post on a webpage, a blog post, a Facebook post, a Twitter tweet or even an updated Facebook status. Text can be obfuscated with a simple cryptography like the Caesar cipher or by a more complex algorithm. The message can be hidden in picture description or in a hidden part of a webpage [13].

To be able to communicate with Botnet every bot has to have predefined communication channels. They have to be integrated into the bot code. A hard-coded list of IP addresses can be easily blacklisted. A stealthier way to maintain the communication channel has to be used to prevent detection of Botnet. Today, DNS techniques can be used as a flexible way of contacting the Botmasters. The most common methods are fast-flux and domain flux [14].

The fast-flux or IP-flux method can be described as rapid and repeated changes of logical address in the DNS record. Even in some cases rapid and repeated changes of logical address of the name server for the domain in a DNS. New name server logical address in most cases implies change of logical address of the host as a new record is used. The rapid change is done by setting a short Time to Live (TTL) for a stored logical address and by changing the appropriate DNS record frequently. This technique is legally used for a long time as a load balancing method for heavy duty servers (web sites and search engines). In the legitimate use, the goal is to distribute the load over multiple servers and to try to dedicate the nearest network server to the client and thus decrease network Round Trip Time.

In Botnets this technique is used to form a Fast-flux service network, a network made by compromised hosts sharing the same domain name. The logical addresses are rapidly swapped based on availability and bandwidth. There can even be a load distribution scheme that employs particular hosts to check the health of other bots in the Botnet. With a rapid change of the logical addresses blacklisting as a technique is obsolete. Recognizing and blacklisting the one percent of the million hosts Botnet has become impossible. One available option is to suspend the domain name, but this is a complex task as most of the Botnets use registrars that are resistant to blocking of domain names [15].

Another advantage of using the Fast-flux is the gained anonymity of the main Botnet command centre. Fast flux bots become proxy server redirecting the traffic and ensuring to disrupt every attempt to track and reveal the identities of the Botnet command centre. This increases a lifetime of the Botnet command centre and makes is easier to set up and manage a compromised network.

To detect Fast-flux and Domain-flux techniques and thus the existence of a bot in a local network a passive or active analysis of the DNS traffic has to be executed. It can be assumed that in the monitored traffic, some of the users are infected with malicious content and that some malware components will be running on monitored systems. These components are likely to contact malicious domains. Malicious domains can be distinguished by studying public malware domain lists and spam blacklists. The goal is to detect patterns in DNS traffic that can be redirected to Intrusion Detection System or Intrusion Prevention System to detect and disable bots in local networks [18].

In active and passive DNS monitoring, multiple features can be monitored. In most active monitoring systems, a Botmaster can detect an attempt to detect a bot. Passive monitoring mostly requires addition time and records to make a decision. Every feature can provide additional information in making a decision about bot existence. A difficult part is to create an algorithm to join multiple feature measurements and to provide a detection threshold to make a correct decision. Some features can be obstructed by Botmaster in adapting bot behaviour. As the detection mechanism is most likely to be a part of the embedded system like IDS system or plug in on a router, performance becomes an issue because of limited processing power and memory availability. Due to these issues, a focus was made on features that can be classified upon the possibility of real time usage and by processing power and memory usage [16=18].

In this paper, Botnet adaptation issues are explored and the most frequently used features were measured. After this step, the best features were selected according to their behaviour, real time performance and possibility of application in embedded systems. An additional new feature is proposed to improve the Botnet detection success. The solution is verified on known inputs: legitimate sites and Botnets.

The remainder of the paper is organized as follows; the related work in this field is presented in the next section. Section 3 presents the overview of the online bot detection features based on document fetch delay. A proposed new feature based on search engines behaviour is introduced in Section 4. Section 5 provides a classification algorithm used for decision about bot existence with actual data examples. Finally, Section 6 describes conclusion and future work.

2 RELATED WORK

In first paper in this field T. Holz et al. presented characteristics of Fast-flux service network and proposed a first technique to detect a bot in a local network [19]. The paper proposed a method of weighed linear regression to assign a flux-score to a domain. The regression function is a linear and made up from three components derived from the DNS: number of unique *A* records, number of a distinct autonomous system numbers (ASN) and number of unique NS records.

E.Passerini et al. developed a program FLUXOR that uses nine features to detect the existence of the compromised network [20]. The nine features are: domain age, place of registration of the domain (country), number of unique records address, time of life Namespace records (TTL), the number of different networks, the number of different autonomous systems, the number of different full domain names obtained by the inverse query, the number of different organization owner and a number of different network names assigned by the registry. The system consists of three modules: the collector, the discoverer and supervisor. Some features are not applicable in the real time and thus the system cannot make instantaneous or quick decisions.

R. Perdisci and other suggests a passive network monitoring through the reduction of traffic observed within a time period [21]. The features that are used are: the number of unique records address, domain name, time of life of the namespace record, the diversity of the individual network prefix, the number of domains in the network, the rate of growth of unique network addresses, diversity of the autonomous system, diversity of the prefix routing protocol (BGP), diversity of the organization which belongs to a network, diversity of network at origin country, the number of dynamic IP addresses (prefix ADSL, DIAL-UP), availability on the network. The same author [22] also publishes similar paper with different features (DNS time to live, frequency of IP change, number of resolved IP, scattered IP over several networks).

J. Nazario and T. Holz continued the study further by combining previously established features (the number of unique record address, diversity of network records and diversity of autonomous system) with new features - the defined retry time in the authoritative server in different autonomous systems [23]. Additional inquiries enabled the visibility of the research by the compromised network. Moreover, in their work it is possible to adapt the network to use a smaller number of different servers.

X. Hu, M. and K. Shin Knysz developed the tool called Digger which may determine the existence of a network of compromised computers [24]. In their work, they rely on features such as the number of unique addresses, the rate of growth in the number of unique records addresses, time of availability of the network address, and the overlap between the network address between domains. Due to the analysis of the growth rate as well as the determination of overlapping addresses in the domain, these features are not usable in real time.

X, Hu, M. Knysz and K. Shinn developed a tool called "RB-seeker" [25]. Their tool uses the following features: the number of unique records address, the rate of growth in the number of unique addresses, the diversity of autonomous systems, the rate of growth in the number of different autonomous systems and inverse name associated with the network address. Based on analysis of characteristics that are based on the grow rate it is evident that the system cannot work in real time. Furthermore, additional tests can alert the network of compromised computers.

S. Marchal, et al. in their work suggest the following features: the number of unique records address, the rate of growth in the number of unique addresses, time of life of the records, the number of unique records of name servers address as well as additional features related to the domain name: similarity of the domain names with the dictionary, the similarity of certain elements of the domain with valid domain names [26]. The work has proposed a new concept for the evaluation of the domain name. Unfortunately, a dictionary must be provided for all languages and additional words should be entered to cover novel computer terms.

L. Bilgi, E. Kirda, C. Krueger, M. Balduzzi propose a tool called Exposure [27]. Exposure is using the following features: availability of a domain, daily similarity in behaviour, repetitive patterns of behaviour and the frequency of access, number of different addresses, different countries for these addresses, the number of different domains with which they share the same address, an average lifetime on name server, standard deviation of the lifetime between addresses, the count of numbers in the domain name and the percentage of the length of the longest part of the domain name with meaning. The system cannot work in real time as it uses comparisons with the past data points.

L. Jehyun, K. Jonghun, S. Hyo-Jeongand L. Hijo propose an analysis of network address and grouping of links addresses with domains [28]. This technique requires an analysis of the data obtained in the past as well as high traffic out to identify the similarities between the behaviour of the observed domain.

H. Ching-Hsiang, H. Chun Ying and C. Kuan-Ta introduce a new feature based on measurement of response time of the domain [29]. A characteristic feature is applicable in real time and does not require significant resources for memory and processing power. There is a problem with legitimate domains, which have no needs to achieve fast response in local segments where it is installed. For example, a local significant portal response time achieved from a distant geographical location.

S. Yadav, et al. analyses DNS traffic by looking for patterns inherent to domain names that are generated algorithmically, in contrast to those generated by humans. Distribution of alphanumeric characters and bigrams mapped to the same set of IP addresses is analysed [30]. This method cannot be used in real time as it is based on statistical analysis. In addition, there is a problem with CDN as they are also using similar domain names wide across the Internet.

O. Pomorova, et al. propose the method that takes into account abnormal behaviours of the hosts' group, which are similar to Botnets: hosts' group has small DNS TTL and carry out the DNS-queries to non-local DNS-servers. The method monitors a large number of empty DNS-responses with error code [31]. Using this technique, there is a high rate of success. However, the method uses a complex algorithm not suited for embedded system. Furthermore, there is a problem when compromised computer detects a Mothership in a few attempts, which is available in a longer period. In that case, a Botnet cannot be detected.

Kwon J. et al. propose a complex system that analyses DNS traffic more systematically by scaling it down. A tool called PsyBoG is developed for detecting malicious behaviour within large volumes of DNS traffic [33]. PsyBoG leverages a signal processing technique, power spectral density (PSD) analysis, to discover the major frequencies resulting from the periodic DNS queries of Botnets. This solution is not applicable in real time as it requires post processing. Additionally, traffic patterns have to be isolated to verify the correct Botnet behaviour in the network. Tab. 1 presents the Botnet detection feature references. For each feature additional fields present applicability in real time, availability to detect examination, to adapt to avoid detection, processing and memory requirements in embedded system environment is identified.

Real time applicability in first column is described as suitable (Yes), can be achieved (Can) and impossible (No). Second and third columns describe Botmaster availability to detect and to adapt as fully adoptable or detectable (Yes), can be achieved in small portion (Can) and undetectable or unadaptable (No). Last two columns classify embedded system requirements as small (Small), medium (Medium) and large (Large) in a sense of processing power and memory requirements.

3 EVALUATION OF FEATURE CHARACTERISTICS

To estimate different features described in Tab. 1 an experimental evaluation was accomplished using diverse sources for domain names.

The first source represents legitimate domains that are the most common in general use. For this purpose, we used top 500 mostly used domains according to Alexa top

sites web page [34]. This set of data is collected during the one-year period and it is referred as D_{CDN}.

Well-known database of Botnet inflicted domains was the second source. As this source, fast-flux Botnets and phish domains were used according to gathered data on Botnet trackers during one-year period [35-47]. The second source set of data is referred to as D_{BOT}.

Additionally, we used internal IDS logs from University of Applied Sciences, Zagreb. IDL logs collected in a period of one year were pre-processed and stored in the database. These logs provided the mostly used locally significant domain names. This set of data was manually classified to provide D_{CDN}' and D_{BOT}' sets.

Table 1 Botnet detection features

	Feature	*Real Time*	*Detect examination*	*Avoid Detection*	*Processing requirements*	*Memory Requirements*
F1	Availability of the domain in the time period [20, 21, 27]	No	No	No	Medium	Large
F2	Place of domain registration (country) [20]	Yes	No	Yes	Small	Small
F3	Number of subdomains in the domain [26]	Yes	No	Yes	Medium	Large
F4	The domain name according to dictionary [26, 30, 32]	Yes	No	Can	Large	Small
F5	The similarity of certain elements of the domain name with a valid domain name [26, 30, 33]	Yes	No	Yes	Medium	Large
F6	Numbers in domain names [27]	Yes	No	Yes	Small	Small
F7	The length of the longest word in the domain name [27]	Yes	No	Yes	Small	Small
F8	Number and duration of the connection [26]	No	No	Yes	Large	Large
F9	Similar daily behavior of the domain [27, 32]	No	No	Yes	Large	Large
F10	Recurring cycles of query to the authoritative server [28]	No	No	Yes	Large	Large
F11	Number of unique address records [19-27, 33]	Yes	No	Yes	Small	Small
F12	The rate of growth in the number of unique network address [21, 22, 24, 25, 33]	Can	No	Can	Medium	Large
F13	Time of life for the network address in name server [24, 31]	Can	No	Can	Medium	Large
F14	Diversity of network address for returned records on request [20-23, 31, 33]	Yes	No	No	Small	Small
F15	The diversity of autonomous system returned from query [19, 20, 21, 23, 25]	No	No	No	Medium	Small
F16	The rate of growth in the number of different autonomous systems [25]	Can	No	No	Medium	Small
F17	A number of different full domain names obtained on request for inverse network address [20, 21, 27]	Yes	Yes	No	Medium	Small
F18	Global name given for registering autonomous system [20]	Yes	No	Can	Medium	Small
F19	A number of different organizations that owns the network [20]	Yes	No	Can	Medium	Small
F20	Number of different countries which belongs to the network [21, 27]	Yes	No	Can	Medium	Small
F21	Reverse name for the network address [21, 25]	Yes	Yes	Can	Medium	Large
F22	Allowed retry time in name server [23]	Yes	No	Can	Small	Small
F23	Time of life for the record in name server [20-23, 26, 27, 31]	Yes	No	Can	Small	Small
F24	Network latency domain and document retrieval time [29]	Yes	No	No	Medium	Small
F25	Calculation of processing time on the server [29]	Yes	Yes	No	Medium	Small
F26	The ratio of network delay and processing time on the server [29]	Yes	Yes	No	Medium	Small
F27	Number of unique records address of the name servers [19, 23, 26]	Can	Yes	Can	Medium	Medium
F28	Number of different autonomous systems address of the name servers [23]	Can	Yes	Can	Medium	Medium
F29	The overlap between the network address and the name server address [24]	Yes	Yes	No	Medium	Medium

The summary set of data used for experimental setup is given by:

$$D = D_{CDN} + D_{BOT} + D'_{CDN} + D'_{BOT} \quad (1)$$

The goal of experimental evaluation was to determine suitable features for embedded system implementation. The aim is to achieve as little processing power and memory needed for implementation into the embedded system.

Tool set was developed to measure all features from Tab. 1 daily. Every feature described in Tab. 1 was implemented as a separate tool. Additional tool was developed to upgrade the sources whose execution was a prerequisite for other tools. Toolset was started daily during a period of one year. Obtained results were stored in database and analysed after the experiment evaluation period has ended.

Measurements have shown that the best results in proportion to the feasibility of real-time and embedded environment implementation advantages has a feature F26 -the ratio of network delay and processing time on the server according to Tab. 1 [29].

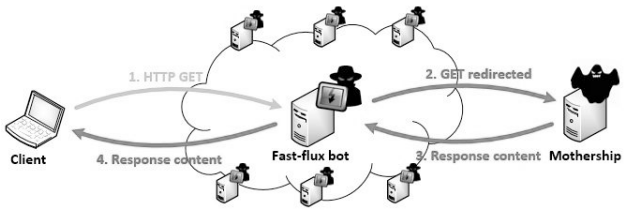

Figure 1 Fast-flux communication between client and the mothership

In fast-flux Botnets the most important concept is the protection of the main command server from disclosure. By discovering and blocking the main command server the whole network of compromised computers becomes non-operational. To prevent this, Botnet network architecture relies on the existence of a large number of proxy servers (i.e. intermediary computers) that are a bridge in communication between one or more main command servers and the compromised client. Fig. 1 presents an example of proxy server communication between client and the mothership. Client sends a request to the proxy server that creates a new communication to the mothership. Mothership is under the control of Botmaster. The role of the mothership is to manage the compromised computers and decide which content and

when is to be served to the clients. When mothership answers, the response is forwarded to the client.

In case of Content Distribution Networks, which represent the legitimate domain, the concept of reliable serving architecture is the same as in Botnets. However, the main computer server has more computational power with faster response times on a faster network, and in most cases optimized for the network on which it advertises. In this scenario, the domain server advertises a unique address that is optimized for the location of the request.

In Botnets optimization for the location is hard to achieve and even in some cases not possible. Botnet proxy computers are primarily selected according to stability, availability on the network and not by the response time of the local network. In [29] H. Ching-Hsiang, H. Chun-Ying and C. Kuan-Ta propose a comparison of the ratio of document fetch time on correct upper layer protocol request, and document fetch time on incorrect upper layer protocol request to detect the existence of Botnet in real-time. Fig. 2 presents structure of estimated network delay.

Estimated network delay is defined as the time needed for a minimal low level communication. In the communication protocol (TCP) client sends a request (SYN packet) for communication to the server. The server responds confirming receipt of the request (SYN ACK). In other words, estimated network delay is defined as the time from sending the request to the receipt of the request [29].

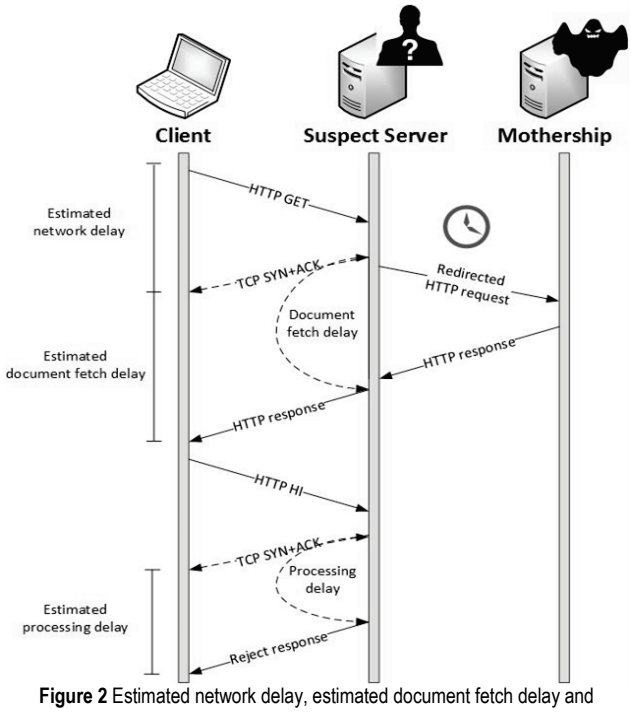

Figure 2 Estimated network delay, estimated document fetch delay and estimated processing delay

Opposite to estimated network delay on low level protocol, estimated document delay on high level protocol is larger. This time is defined as the time from when an initial high level protocol request for communication is sent to retrieve web pages, as in HTTP GET request, (SYN request, followed by the higher layers of the HTTP GET request) and the time when the response came for communication to the higher layer protocol (HTTP response). This time delay is subtracted by the estimated network delay to obtain an estimated document fetch delay.

To determine the distance between proxy server and the main computer server, a deliberate incorrect application request is sent in a higher protocol. It this case HTTP HI is used instead of HTTP GET. Proxy server will recognize faulty request in a higher protocol and will refuse the connection without referring it to the main server. It is assumed that in case of compromised network proxy, computers would be slow (link and processing power) and the distance between an intermediary computer and the host server will not be optimized. If both these parameters are increasing, it is safe to assume that a request is made on the compromised network. Conversely, if both parameters incline to zero then it is the indication of a CDN.

Feature F26 requires additional request in a form of an incorrect higher protocol request be sent to the network domain under examination. This request can be recognized by the Botnet administrator and logged on the administrator owned proxy server. To avoid this, a random generated incorrect request words can be used and, in addition, a database can be employed to exhaust historical results to avoid new queries. Other information, such as network latency and document retrieval time, is obtained by measuring the activity on the link. However, these activities are not visible to the Botnet administrator.

The advantage of using feature F26 is the inability to avoid detection by interfering in Botnet network. When choosing a proxy server in a Botnet the emphasis is on stability and availability. It is not necessary to improve response time from all peers on a local network. Even if the compromised network computers began to optimize the response speed, the number of available stable compromised computer cannot guarantee the response time that can be achieved in optimized CDN.

The disadvantage of this feature is the similar behaviour for networks that have no interest for some regional area. For example, some specific newspaper domain does not have to be optimized for speed in geographically distant domains. This site can use CDN and thus a Botnet can be recognized. The site will have multiple network addresses and for all addresses, a slow connection for proxy server is determined. It is expected that a percentage of these errors will be small. In the future, we expect an increase in a number of CDN and their nodes to further enable reliable operation and optimization in connectivity with local networks [29]. Our experimental evaluation was conducted on the Croatian local network. Due to this reason, the percentage of these errors is not to be neglected.

In Fig. 3 measurement of feature 26 is presented. The horizontal axis represents an estimated document fetch delay in milliseconds, while the vertical axis is an estimated processing delay also in milliseconds. Rounded grey points are known CDN domains and cross black points are Botnets or phishing domains. Tab. 2 presents minimal, average and maximal data point presented in Fig. 3.

There is a significant overlap in dot positioning. In Fig. 3 most of the presented wrong classifications are

based on lack of optimization for CDN domain web site due to the uninteresting market. These web sites can be unavailable to this market or not available in native language and thus, a lack of speed optimization is to be expected.

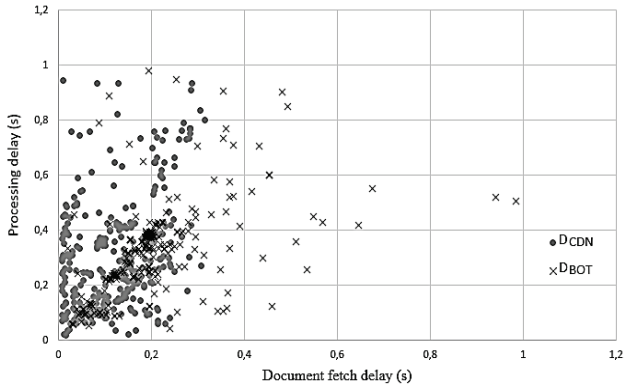

Figure 3 Ratio of estimated document fetch delay with estimated processing delay (F26)

Table 2 Minimum, average and maximum points in Fig. 3

Point	CDN	Botnet
Minimum	(0,006, 0,021)	(0,020, 0,041)
Average	(0,119, 0,292)	(0,211, 0,330)
Maximum	(0,313, 0,947)	(0,984, 0,981)

4 CLASSIFICATION IMPROVEMENT BASED ON SEARCH HITS

To improve the classification of results by decreasing the error rate for a particular local area a new feature is proposed.

Feature F4 empowers corrections based on the domain name by performing a similarity check against the database of known words from a controlled dictionary [26]. This feature has several significant shortcomings. In an embedded system implementation, a large memory is needed to store the dictionary of all known words. As domain names are not related only to the English language, other dictionaries also must be available. Another issue that is not covered by dictionaries is specific words or abbreviations like *"www", "eBay"*, *"PayPal"* although they are very common as legitimate domains in the Internet domain. Thus, dictionaries should be expanded to accommodate such abbreviations.

Search engines clearly recognize different domain names and words from domain names that have significance in today's computing world. They support multiple languages as all countries and languages together create a search tree. For example, word *"eBay"* in most dictionaries will be unknown, but search engines will result in a large number of hits. However, in some countries, *eBay* is not available as a stand-alone site, and thus the domain speed optimization is not a concern to clients.

Utilizing ideas covered by F4 concept, a new feature based on the domain name according to search engines is proposed. This new feature named search engines credit worthiness is referred as F30. This feature is based on search engine hits returned by a domain name to establish if behind the domain is a Botnet. This new feature is used to improve classification introduced by F26 in Fig. 3.

Every domain name is built with words separated by the full stop. The last word describes the largest tree part where this domain resides. This part of the domain is from the pool of available branches in DNS tree. Every word from this pool can be assigned to the compromised or regular domain. However, there are words in this pool that have greater potential to be used in the compromised domain. An example would be a country where digital criminal acts are not regulated and the prosecution of this crime is rare. An algorithm for the feature F30 should eliminate common last word used in regular domains.

The first word in the domain name can also be common as it often describes a service that is behind a domain name. Examples of the common first words would be: www, shop, webmail. Due to this requirement, the feature F30 should eliminate common first words.

To achieve elimination of the common first and last word two new sets of known words are created: *P* (Prefix) and *S* (Suffix). *P* and *S* are created upon the database of known regular sites. An example would be an Alexa top sites list [30]. These domain names are introduced as set L.

There are m domain names verified as regular domain $L=\{L_1,...,L_x,...,L_m\}$. For every Domain name from a set L_x, a list of words is created, where *n* is the length of the domain in words:

$$L_X = \sum_{j=0}^{n} l_{x,j} \tag{2}$$

A list of known regular prefixes $P'=\{l_{\{0,0\}}...l_{\{x,0\}}...l_{\{m,0\}}]$ and suffixes $S'=\{l_{\{0,n\}}...l_{\{x,n\}}...l_{\{m,n\}}]$ is created by repeating (1). These sets are divided into subsets:

$$P' = \bigcup_{i=0}^{u} P_i'' \wedge \bigcap_{i=0}^{u} P_i'' = \phi \tag{3}$$

$$S' = \bigcup_{i=0}^{v} S_i'' \wedge \bigcap_{i=0}^{v} S_i'' = \phi \tag{4}$$

Every subset contains the same elements and all the intersections between subsets make an empty set (3) (4). As there are common words it is expected that $u<<m$ and $v<<m$.

Cardinality for every subset is calculated:

$$\left|P_0''\right|,...,\left|P_x''\right|,...,\left|P_u''\right| \tag{5}$$

$$\left|S_0''\right|,...,\left|S_x''\right|,...,\left|S_u''\right| \tag{6}$$

An average cardinality is calculated for sets P'' and S'': $|P''|_{avg}$ and $|S''|_{avg}$.

Finally, sets *P* and *S* are constructed from words (one element in subset) that are in subsets that have larger cardinality than average cardinality of all subsets. It is expected that cardinality of the *P* and *S* set would be smaller than *u* and *v*.

To measure F30 an algorithm is implemented. Domain data is used from a conducted experimental

evaluation described in Section 3. For every Domain D, a list of word d_n is created:

$$D = \sum_{j=0}^{n} d_i \tag{7}$$

First element d_0 is compared to every element of set P. If P contains element d_0 then d_0 is removed from a set D:

$$\{d_0\} \in P \rightarrow D \setminus \{d_0\} \tag{8}$$

Last element d_n is compared to every element of set S. If S contains element d_n then d_n is removed from set D:

$$\{d_n\} \in S \rightarrow D \setminus \{d_n\} \tag{9}$$

For each remaining word in set D (d_i) a credit worthiness R is examined in the search engine. The final result is an arithmetic mean between all results:

$$R = \frac{1}{n}\sum_{i=0}^{n} d_i, R = R_{CDN} \cup R_{BOT} \tag{10}$$

If the result R is greater than an estimated detection threshold, the domain is declared normal. If the amount is smaller, domain is declared potentially compromised. Threshold can be determined by introducing learning set of domains R_{CDN} and R_{BOT} according to known classification from sources introduced in Section 3. Fig. 4 presents credit worthiness R of the domain according to data source: D_{CND} and D_{BOT}. The detection threshold determination is discussed later in Section 5. X-axis shows the arithmetic mean of all the words in the domain (R) and y-axis is the number of domains that have similar result in percentage. The grey line corresponds to known CDN domains, the black line represents known Botnets, fast-flux Botnets and known phishing sites. As a range of data for the result R is excessive, the Y-axis is presented in logarithmic scale.

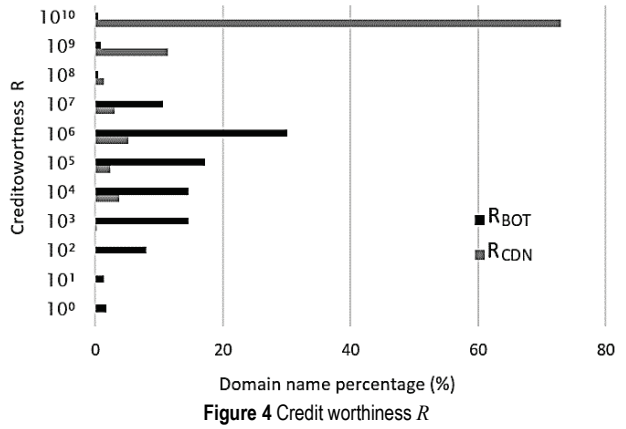

Figure 4 Credit worthiness R

Detailed inspection of credit worthiness shows that certain common domains deviate from the expected credit worthiness of search engines. In this case, it is hard to determine a stable detection threshold value to positively detect Botnets. Measurement results were used and all regular domains that reside next to the proposed detection threshold were examined to improve classification. These results are creating a false negative pattern in F30. Additional intervention is needed to decrease false negative results. Inspection of obtained results revealed that border domain which triggers improper classification could be narrowed into three groups:

1) Companies that are not primarily oriented on Internet;
2) Domains that are not primarily used for browsing;
3) Adult content domain.

Companies like a marketing agency are oriented to their businesses and they used website to present their portfolio to the future client. They offer list of services, recommendations and a contact page. This kind of specialized sites is not visited by a large number of visitors. However, most search engines will recognize them as a legitimate company and offer a contact info on a search page or "more results on" link under the search result. An example would be *www.glipsa.com* representing a renowned marketing agency bit the term *"glispa"* provides less than 200000 hits.

There are domains that are used to offer special services to the clients. Services are served through the application programming interface offered on domain (web services) [48]. An example might be domain *www.mycalendarbook.com* that provides services to other sites or mobile devices while using the network with trusted serving to make API available and load balanced. The website is very simple and thereby provides low credit worthiness to a search engine.

These are the usual sites with legitimate content that search engines intentionally degrade [49]. Such domain names yield poorer results compared to terms that are not linked to adult content sites. Some search engines mark these results as adult content. When this marking is not available it is possible to rely on Meta tag of the site. Adult content page uses a specific Meta tag that indicates that it hosts adult content [50].

To further distinguish results and provide a better detection threshold between regular and compromised domain false negative sites must be eliminated. These are the usual sites with legitimate content that search engines intentionally degrade. Such domain names yield inferior results compared to terms that are not linked to. The answer lies in additional observation gained from a search engine:

1) Search engine offers a contact data about company
2) Search engine offers "more results…" about site
3) Search engine declares "adult content" or an adequate meta tag is found

Indication that domain is a legitimate site can be if Google has recognized a company that is behind these sites and search results offer a small box next to results with the company details (owner, address, working hours, street view) [51]. If these data are provided introduced variable named `company box recognized` -X_c becomes 1.

If a link "more results for ..." appear in webpage with the partitioning of main domain into sections, we could say that this domain is complex or recognized by the search engine [52]. This verifies existence of a domain

name for a longer period. If this data is provided a variable named long domain existence recognized X_r becomes 1.

Some search engines mark adult content due to parenting guidance policy. In lack of these markings, a site page is loaded and searched for tag "adult content" or over 18 warnings [50]. Such sites are intentionally degraded on search engines. If such tags are indeed present a variable named `adult content recognized` -X_p becomes 1.

By introducing additional data in decision conclusion F30 becomes enhanced. The result gained by the search engine hits is corrected by a factor that employs additional information obtained from search engine:

$$R_E = R + (X_c * F_c) + (X_r * F_r) + (X_p * F_p) \quad (11)$$

Factors F_c, F_r, F_p are determined by using a known set of regular domains. In this paper by using known set of learning domains these factors are determined to be 10.

In Fig. 5 enhanced credit worthiness R_E is presented. Upon applying correction factors F_c, F_r, F_p an improved classification is obtained between CDN and Botnet domains.

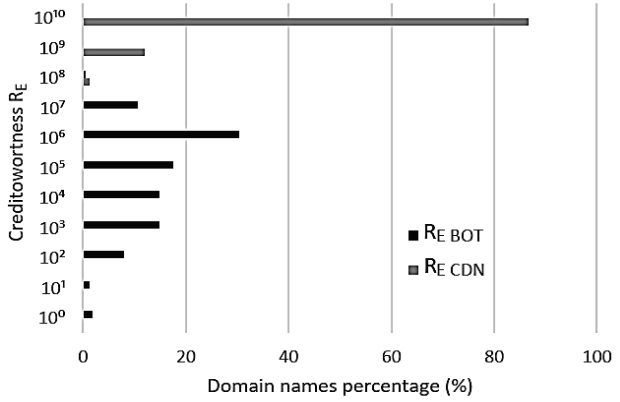

Figure 5 Enhanced credit worthiness R_E

The proposed algorithm is appropriate for implementation in embedded system environment because it can be executed in real time without the necessity of additional memory like a dictionary or black list of words.. Additional calls to search engines and result examination require medium processing power.

The search engine testing is not visible by the Botnet administrator. Subsequently, Botnet administrator cannot verify that a detection is in progress. As search engines have their own databases maintained by specific spider routines, there is no possibility for Botnet administrators to significantly improve the domain hits. Small local domains just by longer existence would improve their hits. For example, other users will mention the domain on a forum, blog or social network and number of hits will rise. Since Botnet domains are new, they are quickly extinguished, rarely mentioned on blogs, forums and social networks and thus the number of hits will be low.

To make a proper classification a detection threshold should be determined. Using the data from the survey detection threshold can be determined since for each output (the number of hits that the search engine is a one-dimensional number), a decision is obtained from a learning set. Ideal detection threshold could be found if the sets could be classifiable, since in such cases the detection threshold can be clearly defined.

However, it is likely to expect that sets will not be directly classifiable. The problem of the classification error is that they are either positive or negative. Positive error is if the CDN domain is recognized as a Botnet. The negative error is if the Botnet is recognized as CDN.

To determine the detection threshold, it is possible to apply the following solutions: 1) set the detection threshold so that there is an equal number of the positive and negative errors, or 2) set the detection threshold so that there are only positive or only negative errors. Positive errors are less problematic than negative errors.

The safest solution is to set detection threshold over the last recognized Botnet. Fig 6. presents details of the critical region where detection threshold is to be determined. There exists an overlapping area between points A and B that covers both CDNs and Botnets. It is necessary to choose the detection threshold to point B just above the last recognized Botnet. By selecting this detection threshold, a positive error is noted: 0.49% compared to the number of domains that we qualify in our experimental setup.

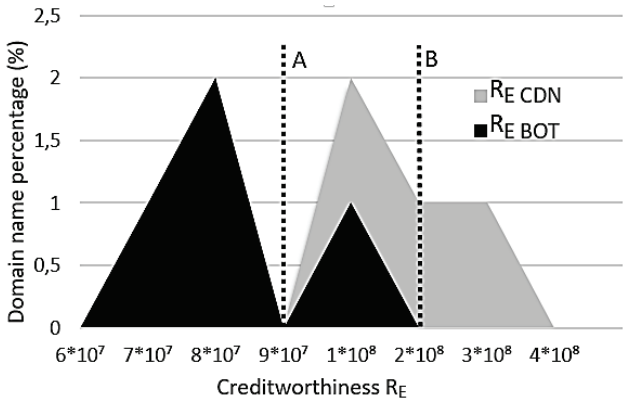

Figure 6 Overlapping CDN and Botnet detection. Enlarged portion of Fig. 5 presenting enhanced credit worthiness R_E.

In a critical case, a Botnet can adapt its behaviour to bluff the results. It is possible only to the certain extent. In most registries, a Botnet will not be able to register a name that is very similar to a frequently used name (e.g. *Paypol*). Even in the event of successful registration small insignificant change in name affects significant influence on credit worthiness obtained by the search engine (e.g. *PayPal* in Google: 540 million, Bing: 19.5 million; *Paypol* in Google: 29.800, Bing: 90.400).

It can be concluded that enhanced F30 cannot positively decide whether the domain is a part of Botnet. This is expected as none of the features described in Table I can make autonomous decision. To make a decision multiple features have to be combined as stated in related work section.

5 THRESHOLD DECISION ALGORITHM

Neither F26 nor enhanced F30 are able to successfully classify CDN from Botnet. To be able to make classification a new approach is suggested. Decision

algorithm is proposed based on usage of F26 and enhanced F30 feature. Feature F26 returns a decimal number representing document retrieval time, with the meaning of the response to a proper request made by higher layer protocol, and processing time as the response to a faulty request made by higher layer protocol. The feature F30 returns decision as binary data. All domains with the number of hits less than the detection threshold value are classified as Botnet controlled domains, while all other domains whose hits are greater than the detection threshold are considered to be hosted by the CDN. Combining these two features improves the classification.

A summarization will increase the decimal number value by the average network delay of 0.25ms for all domains that were declared as Botnet based on the feature F30. For domains that the feature F30 recognized as CDN their decimal values of the feature F26 will be decreased by the value of the average network delay [53].

The data is classified into two sets. The first set is a CDN and the second set is a Botnet. Both sets are processed using Sequential Minimal Optimization algorithm (SMO) to obtain the classification function with Support Vector Machine [53].

Applying the classification function SMO with both sets of data as inputs on WEKA tool [54] a result in a decision function is represented by linear equation:

$$6.1754x - 0.04y = 0.9713 \tag{12}$$

Fig. 7 presents a summarized result of F26 and F30 feature and classification function. WEKA tool successfully calculated unambiguous classification of both sets. Table III presents minimum, average and maximum data points to confirm successful classification.

In Fig. 7 some domains ended in negative values. Since a network latency is contained in the feature F26, subtracting the average network delay significantly affects the domains that were faster in response (i.e. have less delay than the average). Domains that already had an average delay will end around the zero value.

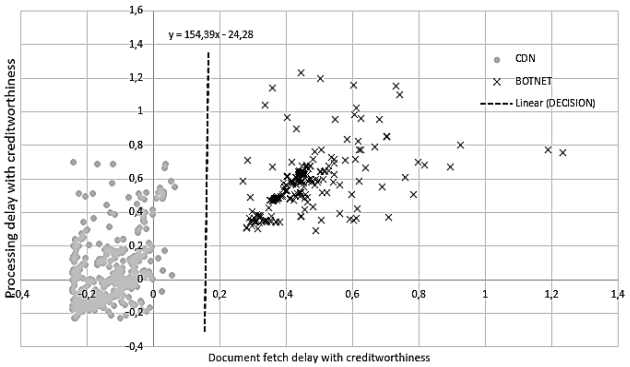

Figure 7 Threshold decision

Table 3 Minimum, average and maximum points in Fig. 7

Point	CDN	Botnet
Minimum	(-0,243, -0,228)	(0,270, 0,291)
Average	(-0,130, 0,042)	(0,461, 0,580)
Maximum	(0,063, 0,697)	(1,234, 1,231)

Critical domains served by CDN are those that are visible on the graph around zero or with a greater value. These domains were very slow in the local network where the measure was taken. There could be several commonplace explanations for such slow response times. For example, these points could represent distant domains that do not have an interest to provide higher speeds in our measured environment.

Classification function presented by (12) need to be adjusted to accommodate a specific period or a device geographic location, since it is expected that the CDN will react different if the local network is changed. To establish a new classification a learning set should be used with known CDN and Botnets. The same learning set can be used regardless of the location of embedded systems since the enhanced F30 adjusts the F26 result. As enhanced F30 uses search engine result it will have the same result undependable of embedded system location. F26 is dependable upon embedded system geographic location as it is based on network delay. Thus a recommendation is to build a new detection threshold upon learning set upon embedded system change location.

If a simple linear function is not possible to be obtained by Sequential Minimal Optimization methods of approximation, a more complex linear function will be yielded. In this case, it is expected that a false positive and false negative error will be introduced. In a case of false positive or false negative, identified problematic nodes should be analysed to determine the cause of wrong classification. For example, the input data declared as CDN picked up from the website that states it as the most common domain can contain domains that serve Botnets. Even the domain served as CDN could be compromised.

6 CONCLUSION

In this paper, a new Botnet detection algorithm is proposed based on two features that are hard to adapt by a Botnet. Document fetch delay detects the proxy existence in document retrieval process. Primary goal of a CDN is to improve the fetch delay to become more available. Contrary, the primary goal of the Botnet is availability. Botnet does not control compromised computers and thus a better candidate for a proxy is compromised computer which was most available in recent time. This makes a CND distinguishable from a Botnet. Unfortunately, some geographic places, which are not of interest to the web site owner, do not require speed optimization. This can lead to a false positive where the CDN domain would be recognized as Botnet.

To improve classification an additional feature should be determined. A solution based on domain name analysis is proposed. Domain name analysis is based on the assumption that legitimate sites are using legitimate dictionary words. Botnets to avoid detection are using very similar words to those in dictionaries. It is difficult for Botnets to use legitimate words because most domains are occupied and the registrars check the names according to their digital security policies. By using an English dictionary our detection algorithm is limited to the English language domain. To introduce additional languages available memory resources and the search efficiency become an issue.

New feature similar to dictionaries based on the search engine credit worthiness is proposed. To distinguish CDN and Botnet domains a domain words

number of search engine hits is analysed. Additionally, feature is improved by analysing additional information about domain obtained from search engine results. Geographic location is no longer an issue as search hits are similar from different locations. Botnets are constantly changing to avoid detection. Botnet domain can hardly become usable and referenced to obtain a large number of hits in search engines and the ownership will most likely be unknown. Experimental evaluation confirmed better classification than the network delay feature. Unfortunately, unique classification was not accomplished. Both features separately cannot classify CDN from Botnet sites. By combining the two features, classification is widely improved. New detection algorithm is proposed combining network delay and domain name credit worthiness. The algorithm advantage is the ability to detect Botnet in real time with a small memory signature and processing performance, which makes it suitable for embedded system. Disadvantage is the threshold assessment, which has to be calculated with a learning set for a particular implementation.

In future, we plan to implement this algorithm into multiple embedded systems set in different geographic locations. The goal is to provide new sets of data from different networks. By comparing these sets, a CDN behaviour for different local networks can be more precisely modelled. Additionally, as there are multiple search engines experimental setup can be conducted in hits differences on words that are common to compromised domains.

Acknowledgements

The authors would like to thank the Central Informatics Support staff of Zagreb, University of Applied Sciences for gathering data.

7 REFERENCES

[1] Mockapetris, P. (1987). Domain names – Concepts and Facilities. Network Working Group RFC-1034. http://tools.ietf.org/html/rfc1034 (01.03.2016) https://doi.org/10.17487/rfc1034

[2] Mockapetris, P. (1987). Domain names – Implementation and Specification. Network Working Group RFC-1035. http://tools.ietf.org/html/rfc1035 (01.03.2016) https://doi.org/10.17487/rfc1035

[3] Brisco, T. (1995). DNS Support for Load Balancing. Network Working Group RFC-1794. http://tools.ietf.org/html/rfc1794 (01.03.2016) https://doi.org/10.17487/rfc1794

[4] Hofmann, M.; Leland, R.B. Content Networking Architecture, Protocols, and Practice. Morgan Kaufmann Publisher, 2005.

[5] Nygren, E., Ramesh, K. S., & Sun, J. (2010). The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS Operating Systems Review, 44*(3). https://doi.org/10.1145/1842733.1842736

[6] Google Public DNS Frequently Asked Questions: Where are your servers currently located?, https://developers.google.com/speed/public-dns/faq, (01.03.2016)

[7] Lee, K. W. (2005). Protecting Content Distribution Networks from Denial of Service Attacks. *IEEE International Conference*, 830-836.

[8] Zlomislić, V., Fertalj, K., & Sruk, V. (2017). Denial of service attacks, defences and research challenges. *Cluster Computing The Journal of Networks, Software Tools and Applications, 20*(1), 1-11. https://doi.org/10.1007/s10586-017-0730-x

[9] Ramneek, P. (2013). Bots & Botnet an Overview (PDF). SANS Institute https://www.sans.org/reading-room/whitepapers/malicious/bots-Botnet-overview-1299 (20.02.2016)

[10] Caglayan, A., Toothaker, M., Drapeau, D., Burke, D., & Eaton, G. (2012). Behavioral analysis of Botnets for threat intelligence. *Information Systems and e-Business Management*, 830-836. https://doi.org/10.1007/s10257-011-0171-7

[11] Qi, L. & Zhen, L. (2014). Portfolio optimization of computer and mobile Botnets. *International Journal of Information Security, 13*(1), 1-14. https://doi.org/10.1007/s10207-013-0206-9

[12] Asghari, H., Eeten, M. J. G., & Bauer, J. M. (2015). Economics of Fighting Botnets: Lessons from a Decade of Mitigation. *IEEE Security & Privacy, 13*(5), 16-23. https://doi.org/10.1109/MSP.2015.110

[13] Anagnostopoulos, M., Kambourakis, K., & Gritzali, S. (2016). New facets of mobile Botnet: architecture and evaluation. *International Journal of Information Security. 15*(5), 455-473. https://doi.org/10.1007/s10207-015-0310-0

[14] Salusky, W. & Danford, R. (2007). Know Your Enemy: Fast-Flux Service Networks Know Your Enemy: Fast-Flux Service Networks. *The Honeypot project*. http://www.honeynet.org/papers/ff (04.02.2016)

[15] Nadji, Y., Antonakakis, M., Perdisci, R., Lee, W., & Dagon, D. (2013). Beheading Hydras: Performing Effective BotnetTakedowns. *20th ACM Conference on Computer and Communications Security, ACM CCS 2013*, 121-132. https://doi.org/10.1145/2508859.2516749

[16] Scarfone, K. & Mell, P. (2010). *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Computer Security Resource Center (National Institute of Standards and Technology).

[17] Alieyan, K., ALmomani, A., & Manasrah, A. (2015). A survey of Botnet detection based on DNS. *Neural Computing and Application*s, 1-18.

[18] Gajski, D. D., Abdi, S., Gerstlauer, A., & Schirner, G. (2009). *Embedded System Design - Modeling, Synthesis and Verification*, Springer US. https://doi.org/10.1007/978-1-4419-0504-8

[19] Holz, T., Gorecki, C., & Rieck, F. (2008). Measuring and detecting fast-flux service networks. *Proceedings of the Network & Distributed System Security Symposium*, 12-22.

[20] Passerini, E., Paleari, R., Martignoni, L., & Bruschi, D. (2008). FluXOR: Detecting and Monitoring Fast-Flux Service Networks. *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 186-206. https://doi.org/10.1007/978-3-540-70542-0_10

[21] Perdisci, R., Corona. I., Dagon, D., & Lee, W. (2009). Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. *Proceedings of the 2009 Annual Computer Security Applications Conference*, 311-320. https://doi.org/10.1109/ACSAC.2009.36

[22] Perdisci, R., Corona I., & Giacinto G. (2012). Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Analysis. *IEEE Transactions on Dependable and Secure Computing, 9*(5), 714-726. https://doi.org/10.1109/TDSC.2012.35

[23] Nazario, J. & Holz, T. (2008). As the net churns: Fast-flux Botnet observations. *MALWARE 2008, 3rd International Conference on Malicious and Unwanted Software*, 24-31. https://doi.org/10.1109/MALWARE.2008.4690854