Scientia Iranica D (2015) 22(6), 2389–2400

SCIENTIA IRANICA

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
www.scientiairanica.com

Detection of fast-flux botnets through DNS traffic analysis

E. Soltanaghaei and M. Kharrazi[*]

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

Received 31 July 2014; received in revised form 2 February 2015; accepted 15 September 2015

KEYWORDS
Botnets;
Bot;
C&C channel;
Fast-flux;
IP-flux,
DNS server.

Abstract. Botnets are networks built up of a large number of bot computers, which provide the attacker with massive resources, such as bandwidth, storage, and processing power, in turn, allowing the attacker to launch massive attacks, such as Distributed Denial of Service (DDoS) attacks, or undertake spamming or phishing campaigns. One of the main approaches for botnet detection is based on monitoring and analyzing DNS query/responses in the network, where botnets make their detection more difficult by using techniques such as fast-fluxing. Moreover, the main challenge in detecting fast-flux botnets arises from their similar behavior with that of legitimate networks, such as CDNs, which employ a round-robin DNS technique. In this paper, we propose a new system to detect fast-flux botnets by passive DNS monitoring. The proposed system first filters out domains seen in historical DNS traces assuming that they are benign. We believe this assumption to be valid as benign domains usually have long lifetime as compared to botnet domains, which are usually short-lived. Hence, CDN domains, which are the main cause of misclassification when looking for malicious fast-flux domains, are removed. Afterwards, a few simple features are calculated to help in properly categorizing the domains in question as either benign or botnet related. The proposed system is evaluated by employing DNS traces from our campus network and encouraging evaluation results are obtained.

1. Introduction

One of the most important and prominent sources of threats on the internet are botnets. These networks are built up of a large number of bot computers which provide the attacker with massive resources such as bandwidth, storage, and processing power. In turn, with such resources, the attacker will have the capability to launch massive attacks such as Distributed Denial of Service (DDoS) attack, or undertake spamming or phishing campaigns. The main characteristic which distinguishes bots from regularly infected machines is their ability to coordinate their action through a Command and Control (C&C) channel under the control of a bot master (i.e., the attacker).

There are two main approaches in detecting botnets. One is based on monitoring and analyzing network traffic, looking for signs that indicate a computer is connected to a C&C server or other behavior indicative of the presence of a bot in the network. The other approach proposed by researchers is based on analyzing DNS query/responses in the network. The importance of such an approach would be better understood considering the fact that bots need to locate the C&C server by resolving its domain. Hence, by monitoring this channel, abnormal DNS query patterns could be detected, potentially before the bot communicates with the C&C server. Each approach has its advantages and disadvantages. The main advantage of network traffic analysis is that it

[*]. *Corresponding author. Tel.: +98 21 66166627*
E-mail addresses: soltanaghaei@ce.sharif.edu (E. Soltanaghaei); kharrazi@sharif.edu (M. Kharrazi)

enables the detection of some types of botnets, such as peer to peer botnets, which cannot be detected by DNS traffic analysis. Nevertheless, analyzing large volumes of network traffic has its own difficulties. Furthermore, if bots are encrypting their communication to the C&C server, then techniques based on analyzing the content of packets sent over the network would not work. This is while DNS traffic cannot be encrypted. Hence, in this work we concentrate on a detection scheme based on monitoring DNS traffic.

Moreover, attackers take different defensive measures to make their botnets more resilient to detection and, in turn, take-down by network providers. One widely used approach is fast-fluxing, in which the domain name for the C&C server resolves multiple IP addresses, leading to different copies of the C&C server. Hence, the traffic of the botnets is distributed among different C&C servers, in turn, making their detection more difficult. The main challenge in detecting fast-flux botnets is that their behavior is very much similar to that of the legitimate networks using the Round-Robin DNS technique (RRDNS) [1]. As shown in [2], due to the importance of availability in internet services, large websites use the RRDNS method to distribute the incoming traffic between different servers. This method is also used in Content Distribution Networks (CDNs) and makes their behavior similar to that of the fast-flux botnets. Therefore, using the RRDNS method by the large CDNs causes difficulty in accurate detection of fast-flux botnets.

In this paper, we introduce a new system to detect malicious fast-flux domains by passive DNS monitoring. The proposed system has two main modules for analyzing the behavior of fast-flux botnets. First, it removes a high percentage of benign domains, including those of the CDNs, with the aid of historical traffic. The main idea employed in this module is the short lifespan of botnet domains, as noted in [3], in comparison with the benign CDN domains. Thus, by eliminating the domains present in historical traffic, the CDN domains will be removed (because of their long lifespan) and malicious domains with short lifespans will remain for further analyses. Furthermore, this paper provides an efficient implementation for the usage of historical traffic as white-list. The second module uses two fundamental attributes of fast-flux domains (high flux of resolved IPs and small TTL values) in the form of probability functions to determine the flux rate for each domain. In summary, the contributions/highlights of this work include:

- This paper presents a novel approach for detection of malicious fast-flux domains. The proposed system calculates the flux rates of a domain by determining the flux probability functions which are based on fundamental characteristics of fast-flux domains (i.e., high fluxing of IPs). As this approach relies on basic features, it can detect different kinds of fast-flux domains without relying on information from a specific family of botnets.
- This paper employs historical traffic as a white-list. Using historical traffic not only reduces the size of traffic being analyzed, but also eliminates the CDN domains which are the main cause of misclassification when looking for malicious fast-flux domains. The Bloom filter data structure is used to store historical traffic with minimal storage space.
- The proposed system adopts Sequential Probability ratio Testing (SPRT) as a statistical method to provide online detection. The SPRT method aggregates the results of flux probability functions of each domain in sequential time windows.
- The proposed system is evaluated by employing DNS traces from our campus network. Furthermore, it is shown that the proposed system can achieve a 94.44% detection rate and 0.001% false positive rate.

The remainder of this paper is organized as follows, Section 2 reviews related works. In Section 3, the proposed system is explained in detail. Section 4 evaluates the performance of the proposed system. Also, it provides a short explanation about the specificities of the system implementation. Results are discussed in Section 5, and the manuscript is concluded in Section 6.

2. Related works

There has been much prior work in the field of botnet detection. A number of approaches monitor and analyze network traffic [4-7], whereas techniques such as those in [8-12] concentrate on DNS traffic. In this paper, we focus on DNS-based detection techniques, which analyze fast-flux botnets, and consider proposals on analyzing domain-fluxing techniques as out of scope.

A group of studies has analyzed the behavior of botnets and their characteristics. Konte et al. [13] show that despite the similarity of the TTL values in fast-flux networks and CDNs, the operations of the two networks are different. For instance, the IP distribution of malicious fast-flux networks has a larger variance than that of the content distribution networks. In 2009, Caglayan et al. [3] observed that fast-flux domains usually have a short lifespan of about 2 weeks to 2 months.

The first fast-flux detection system among active approaches was proposed by Holz et al. [8]. This system calculates a flux score for each domain based on the number of distinct A records and the number of NS records for a domain. One of the weaknesses of this approach is that the features used cannot distinguish fast-flux domains from CDN domains properly, which results in false positives. A number of approaches, such as those in [14,15], have been built on the work of Holz et al., wherein initially suspicious domain names are gathered from spam emails and then those domains are monitored by active probing. The main drawback of such approaches is that they rely on active DNS probing; furthermore, the domains they monitor are limited to those observed in spam emails. Moreover, active probing DNS-based techniques, such as those noted above, create excessive network traffic and may be detected by an attacker.

Perdisci et al. [9] proposed a detection technique based on passive monitoring of DNS traffic without limiting the detection technique to domain names extracted from spam emails and black-lists. They calculated the similarity between the resolved IP sets of different domains in order to cluster domains belonging to the same service, and then, based on a set of active and passive features, classified the clusters as either benign or malicious. Later, Perdisci et al. [10] proposed FluxBuster, which was built on their earlier work, and, in essence, removed the dependency on active features and relied solely on passive features.

Antonakakis et al. [11] proposed Notos, with which reputation scores are assigned dynamically to domain names based on their level of malicious activities. Notos uses the passive DNS analysis and it assigns a low reputation score to the malicious domains according to the network- and zone-based features. The network-based features determine the fluxy behavior of each domain and zone-based features distinguish benign CDN domains from malicious ones. As a limitation, Notos is not able to assign a correct reputation score to the domain names with very little historical information and it requires a large passive DNS collection. In [12], a system called Exposure is introduced which employs a passive DNS analysis technique to detect malicious domains. Exposure tries to detect abrupt changes in the number of requests to the domains which show abnormal behaviors.

Recently, Choi and Lee [16] have proposed a mechanism called botGAD by monitoring malicious group activities in the DNS traffic. To find the group activities, botGAD measures similarity of different domains by building a matrix based on features extracted from DNS traces. However, as the analysis is performed periodically and in independent time windows, botnets can circumvent detection by delaying communication and spreading their traffic over multiple time windows.

Most of the previously proposed techniques could be circumvented by some changes in the activities of bots, like producing noisy traffic (i.e. generating fake DNS queries), sub-grouping members of a botnet, etc. improving the methods of IP assignments. Given the shortcomings present in previous approaches in fast-flux botnet detection, we propose a new detection mechanism based on the fundamental and inherent characteristics of fast-flux botnets. We should note that as the proposed approach analyzes domains seen in requests, independently, it is not affected by the behavior of hosts in the network or other domains queried, as are some of the previously proposed techniques.

3. System overview

One of the main challenges in detecting malicious fast-flux domains is the similarity between such domains and domains employed in CDNs. These legitimate networks distribute the traffic loads of a website between several servers by using the round-robin DNS technique, where a group of IPs are assigned to a domain name and a permutation is returned in response to each request. The used IP sets have small TTL values in order to control the load on each server, while changing their permutation in sequential requests. Similarly, these two features (the large number of resolved IPs and small TTL values) are used in malicious fast-flux networks. Hence, distinguishing malicious domains from benign domains becomes a challenge.

Nevertheless, with a closer examination in [8], it is shown that IP allocation is done differently in CDNs vs. fast-flux networks. In the CDNs, the list of resolved IPs is generally fixed and their permutation is changed as requests come in, while in the fast-flux networks, the list of resolved IPs can be different in each response. The reason is that compromised servers are used for hosting malicious domains and they could get turned off by their owners or cleaned up. Hence, alternate compromised servers are used. In addition, these compromised servers are distributed in different subnets. Therefore, there is no relationship between IPs returned by the fast-flux networks.

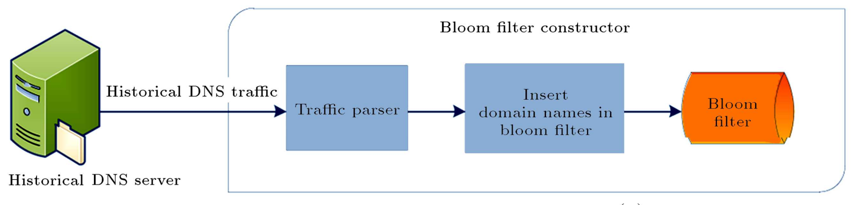
In the rest of this section, the proposed system is discussed in details. The main objective of the proposed method is to detect fast-flux botnet domains. Section 3.1 describes the overall architecture of the proposed system. In Sections 3.2 and 3.3, two features of the system are presented including Bloom filter application in history storage and defining probability functions for showing malicious rate of a domain. Then, Section 3.4 describes the application of SPRT algorithm and how it is employed in the proposed detection technique.

3.1. System architecture

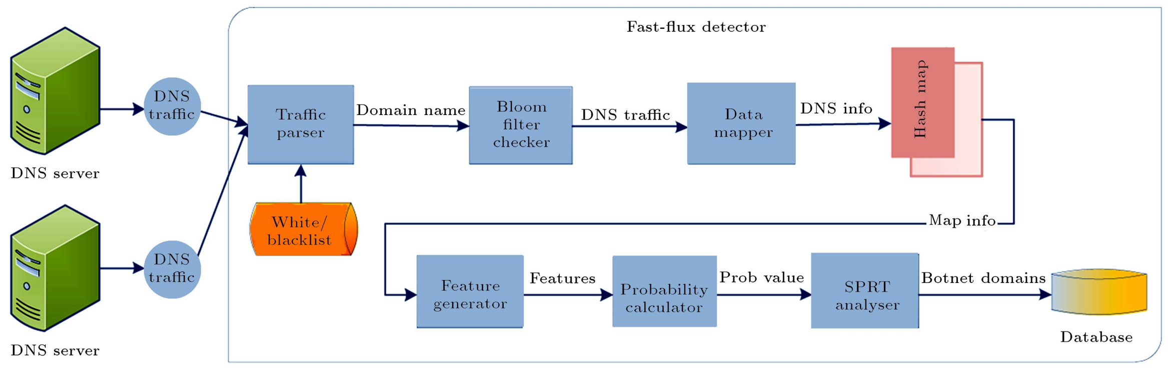
The proposed system uses three basic attributes of malicious fast-flux domains: (1) large number of resolved IPs, (2) small TTL values, and (3) different IP allocation methodology in malicious fast-flux and CDN domains: different IP sets in sequential responses vs. repeating IP sets. The first and third attributes are used in defining flux probability functions and the TTL value is checked as the last filter in the detection module.

More specifically, there are two main modules of (1) storing history, and (2) online traffic analysis with the aim of botnet detection. The system architecture is shown in Figure 1. Figure 1(a) includes two sub-processes of "Traffic Parser" and "History Saver". In the "Traffic Parser", the DNS history traffic is collected and the required information such as domain names are extracted. Afterwards, the "History Saver" stores the extracted domain names within the Bloom filter data structure. The utilization of this data structure will reduce the volume of data stored as well as the search time overhead. (Further details on Bloom filters are described in Section 3.2.)

Figure 1(b) illustrates the detection service components. In this step, the collected DNS traces are parsed and the required information is extracted, i.e. "Traffic Parser". Then the extracted domains are compared with those stored in the Bloom filter, i.e. "Bloom Filter Checker". If a domain name exists in the history, then it does not belong to a malicious IP flux network and can be discarded. The domain names, that do not exist in the Bloom filter, are inserted into the hash map data structure built on top of the data structure proposed in [16]. As shown in Figure 2, the hash map includes a domain map that has a domain name as a key and a requesting IP map



(a)



(b)

Figure 1. The overall architecture of the proposed system

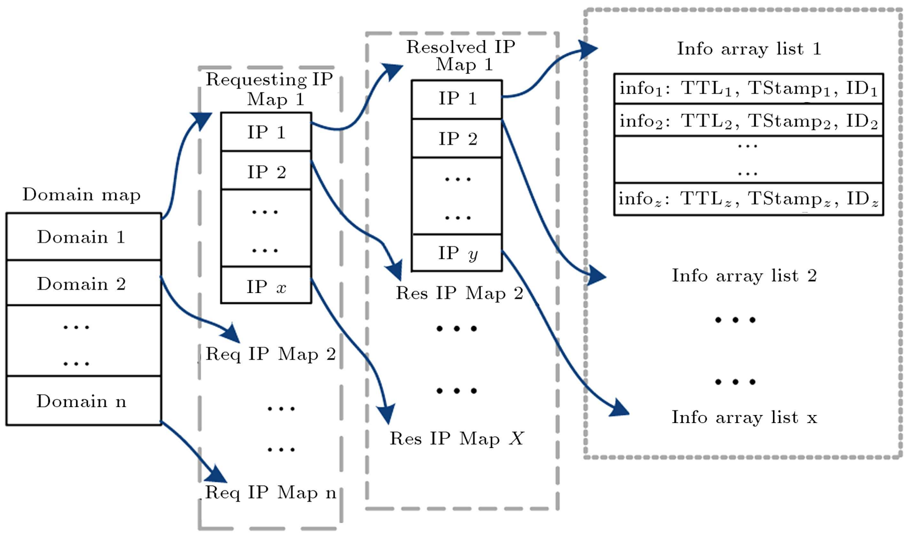


Figure 2. The hashmap data structure.

as a value. Requesting IP maps have an IP address of the requesting host as a key and a resolved IP map as a value. This structure is repeated in the next map, in which the key is the resolved IP address and the information list consists of the TTL and time stamps as values. So, each DNS query constitutes a branch of this tree data structure. In the next step, the "Feature Generator" block extracts the required features. The probability functions measure the amount of resolved IP growth for the domains which is in fact the value of the flux rate. Then, the SPRT analyzer performs a hypothesis test with the aim of online detection of malicious domains. Detected domains are stored in the database and suspicious domains and their value of IP growth are stored in an alternate database to be used in later analysis.

The flowchart of the proposed system is shown in Figure 3. In the first step, the history traffic is stored in the proper data structure. Then, the analyzed traffic is gathered in a preset time window and compared with the white-list. In the next step, the remaining domains are compared with the history. Hence, both domains found in the white-list and history are removed. At the end of a time window, the proper features are extracted and two probabilistic functions are calculated to determine flux rates of a domain name. In the final step, a sequential probability testing is used for making a decision about each domain according to the value of probabilistic functions. This statistical method can produce three different outputs of "malicious", "benign", and "Inadequacy of information" for every

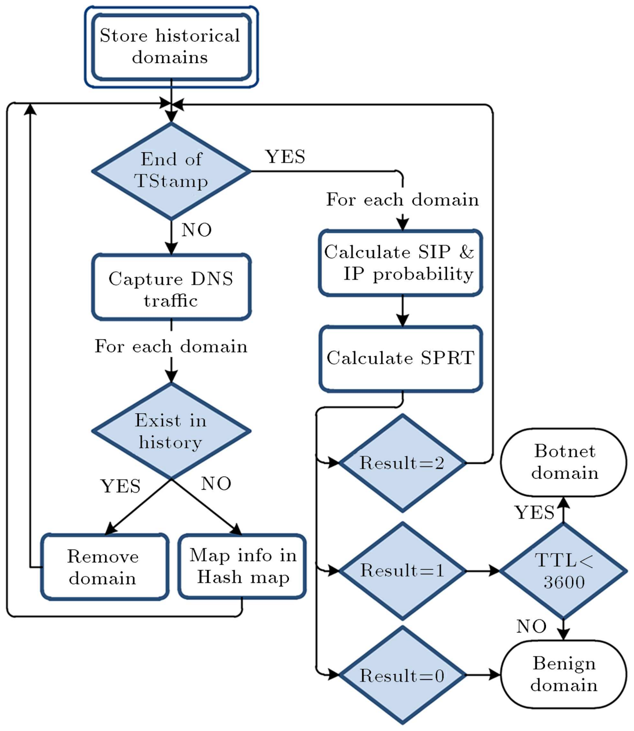


Figure 3. The proposed system flowchart.

analyzed domain. If the information is not sufficient, the domain name will be stored for examination in the next time window. Furthermore, if the output of the sequential testing is malicious, the TTL condition will be checked. Hence, a domain will be recognized as malicious if it has three characteristics of a malicious domain explained earlier in this section.

3.2. *Historical data storage*

The proposed system uses historical traffic to remove benign domain names, including CDNs. But, there are two challenges in using historical traffic for CDN filtering. First, the volume of historical traffic would be quite large and a proper storage methodology would be required. Second, the comparison of current traffic with the large volume of history needs an optimized search algorithm in order to check the existence of each requested domain with minimal time overhead. To solve these challenges, we employed the Bloom filter data structure [17], which is able to provide efficient storage and searching with an adjustable false positive and a false negative error rate of zero.

The Bloom filter data structure works by storing each domain name into a number of bits in a bit array initialized to all zeros. This is done by using one or more hash functions to indicate the related indices in the array which will be set to one. After storing multiple domains, multiple locations in the bit array would have a value of one. In the search phase, the domain being searched is hashed and it is checked if all the corresponding locations in the bit array are set to one; if that is the case, then it can be concluded that the domain in question has been previously stored in the bloom filter.

Furthermore, the Bloom filter data structure provides the ability to determine the size of storage array according to the value of false positive error rate. Where false positives may occur when a set of indices corresponding to a domain are changed to ones, when other domains cause the change at those indices. Moreover, Bloom filters do not have a false negative error rate. In other words, if a domain name is inserted into the storage array, the search results for that domain name will always be positive. This attribute is essential in the performance of the proposed system, because it could guarantee that all the domains present in the history files, including CDNs, will be filtered accurately. This would reduce the overall false positive error rate of the detection system.

3.3. *Flux rate of a domain*

As noted earlier, after comparison of the domain names with history and storing the required information in the hash map, the desired features are extracted in order to calculate the flux rate of the domains. Generally, the flux rate quantifies the change/growth in the resolved

Table 1. Features used.

Row	Feature	Description
1	#SReq	Number of single-IP requests for a domain
2	#MReq	Number of multiple-IP requests for a domain
3	#SResIP	Number of distinct resolved IPs in single-IP type for a domain
4	#MResIP	Number of distinct resolved IPs in multiple-IP type for a domain
5	MReqSize	The average number of resolved-IP in multiple-IP requests of a domain
6	#FirstMResIP	The number of resolved IPs in the first packet of a domain name in a time window
7	TTL value	The average value of time to live of a resolved IP for a domain

Table 2. Examples of DNS single and multiple response.

1	Single domain	www.hamunshop.ir	76.164.198.3
2	Multiple domain	forthworth.biz	216.239.34.21
		forthworth.biz	216.239.36.21
		forthworth.biz	216.239.38.21
		forthworth.biz	216.239.32.21

IPs for each domain. The list of the required features are shown in Table 1. The two main features are the number of requests and the number of distinct resolved IPs for a domain.

The definition of the IP growth of a domain differs based on the number of resolved IPs in each DNS response. As shown in Table 2, DNS response packets can include one or more resolved IPs. If the packet consists of one resolved IP, IP growth can be calculated by the division of the number of distinct resolved IPs to the number of requests. But if the packet includes more than one resolved IP, this definition does not correctly calculate the IP growth rate, because the multitude of IPs in one packet implies false growth. For example, if each DNS response related to a domain contains four different IPs and the same IPs are replied in response to the next DNS request, this domain does not have any IP growth; while based on the previous definition, the IP growth equals to 4 resolved IPs divided by 2 queries or, in other words, 200%, which is not correct. To solve this confusion and as shown in Table 1, we define two IP growth functions for the two groups of *Single-IP* DNS responses and *Multiple-IP* DNS responses.

The flux rate of *single-IP* (noted as "single") responses is defined using Eq. (1). The first factor calculates IP growth based on the number of distinct resolved IPs among all responses and it implicitly shows the flux rate of single responses in a time window. The second factor, as introduced in [9], is a sigmoidal weight that measures the confidence of the IP growth value. Because, while the values of "SResIP" and "SReq" might be small, the IP growth value may still be high. Therefore, this factor reduces the confidence of the IP growth when the resolved IP sets or request sets are small. For example, if #SResIP=3 and #SReq=5 or #SResIP=30 and #SReq=50, the SIP value is 0.6 in both cases, but the confidence in the flux rate value in the second case is higher. The value of γ is considered to be 3 by using the same logic proposed in [9].

$$SIP = \frac{\#SResIP}{\#SReq} \times \frac{1}{1 + e^{(\gamma - min(\#SResIP, \#SReq))}}. \tag{1}$$

Similarly, the IP growth rate of *multiple-IP* (noted as "multiple") responses of a domain is calculated using Eq. (2). The main purpose of this equation is to calculate the average number of new IPs of a domain resolved by each DNS response. To achieve this aim, the resolved IPs of the first DNS response for a domain is stored in a time window and by comparison with next IPs, the number of new IPs in the next responses are measured. This value constructs the numerator of IP growth function. Then, by dividing this value to the number of DNS responses during the time window, the average number of new IPs in each DNS response will be calculated. Therefore, Eq. (3) defines the multiple flux rate of a domain. The first factor calculates the flux rate by dividing the IP growth value to the average size of multiple DNS responses and the second factor is the confidence weight. The best value for γ is considered to be 2 based on the same logic proposed in [9].

$$IP\ Growth = \frac{\#MResIP - \#FirstMResIP}{\#MReq - 1}, \tag{2}$$

$$MIP = \frac{IP\ Growth}{MReqSize} \times \frac{1}{1 + e^{(\gamma - min(MReqSize, \#MReq))}}. \tag{3}$$

3.4. Online detection by sequential testing

Most of the previous detection techniques noted earlier would require long intervals of observed traffic in order to detect a botnet in the network. Our goal in the proposed technique was to limit this time delay, and, when possible, report the existence of a bot in the network. As such, the traffic is analyzed in short time windows and the IP growth is calculated for the requested domains independent from other time windows. Then, the calculated values are combined with those of the previous time windows for each domain in order to improve the confidence of the system output. To that aim, the Sequential Probability Ration Testing (SPRT) is used as a statistical method, with which the current IP growth value of a domain is added to those observed in previous time windows.

The definition of this statistical method, based on the International encyclopedia of statistical science [18], is as follows. There are two hypotheses, denoted as H_0 and H_1, which we consider for each domain under consideration, where they correspond to benign and malicious domains, respectively. For each hypothesis, a Probability Density Function (PDF) is defined like Eq. (4) in which S_1 is a random variable in state 1.

$$f_1(S_1) = K_1, \qquad f_0(S_1) = K_0, \tag{4}$$

K_0 and K_1 denote the probability of a domain being benign or malicious, respectively. In the proposed system, the PDFs of f_0 and f_1 are considered as follows:

$$f_0(S_i) = 1 - SIP(\text{or } MIP),$$
$$f_1(S_i) = SIP(\text{or } MIP). \tag{5}$$

By considering $S_1, S_2, ..., S_n$ as the samples of different time windows, in which the SIP and MIP are calculated, the likelihood ratio will be defined as:

$$\Lambda_n = \frac{f_1(S_1, S_2, ..., S_n)}{f_0(S_1, S_2, ..., S_n)} = \Pi_{i=1}^{n} \left\{ \frac{f_1(S_i)}{f_0(S_i)} \right\}. \tag{6}$$

The equation can be converted to an accumulated form of likelihood ratio for independent analysis of each time window. Then, given two thresholds of T_0 and T_1 where $T_0 < T_1$, at each time window, the likelihood ratio is calculated and it is compared with these thresholds. Therefore:

- If $\Lambda_n \leq T_0$, then H_0 is accepted;
- If $\Lambda_n \geq T_1$, then H_1 is accepted;
- If $T_0 \leq \Lambda_n \geq T_1$, then the current information is not enough to make a decision and the analyses should be continued.

We should note that the above noted thresholds, T_0 and T_1, can be calculated based on:

$$T_0 = \frac{\beta}{1-\alpha}, \qquad T_1 = \frac{1-\beta}{\alpha}, \tag{7}$$

where, α and β are the desired false positive and false negative rates, respectively.

Based on the SPRT function, MIP and SIP values are calculated at the end of each time window for each domain. Then, the likelihood ratios are calculated. If the ratio exceeds one of the thresholds, it means that enough information is provided to decide on a domain and it is assumed as a malicious or benign domain. Domains with likelihood ratios between the two thresholds are saved to be examined in later time windows.

4. Implementation and evaluation

The proposed system was implemented by integrating a set of modules implemented in C, Java, and Shell scripts on a Linux operating system. Additionally, a set of open source tools, such as Rapidminer [19] and Tshark [20], were employed. Furthermore, the system was deployed at the Sharif University of Technology campus, gathering traffic from the primary DNS servers of the campus. In what follows, we will first introduce the dataset used and the evaluation methodology employed in Section 4.1 and then evaluate effectiveness of the bloom filters in Section 4.2. Detection accuracy of the proposed system is studied in Section 4.3.

4.1. Datasets and evaluation methodology

We have used two sets of traces in order to evaluate the proposed botnet detection technique. These traces are:

- Trace #1: Multiple DNS traces obtained from the Sharif University of Technology primary DNS server (i) from March 1st through 30th, 2012 and (ii) from January 1st through 31st, 2013.
- Trace #2: DNS traces were collected while executing three botnet samples on a VM for 3 days within a controlled environment; at the same time, a number of benign applications were also executed on the same VM. Afterwards, the collected DNS traffic was merged with the DNS traces collected at the campus level between January 1st through 31st, 2013.

For our historical dataset, we employed DNS traces collected from January 1st through December 31st, 2011. Furthermore, we used a white-list to remove well known domains (i.e. google, yahoo, etc.) from the dataset. The white-list was created based on the Alexa's ranking [21] and by selecting the top 100 popular domains visited by clients. Furthermore, some of domains, we thought could be potentially employed by a botnet (e.g. domains, related to weblog services), were removed from the white-list.

To evaluate accuracy of the proposed technique, we needed to label the domains found in the DNS traces as either benign or malicious. To that end, a black-list was created from several sources, which includes the MalwareDomainList [22], DNS Blackhole [23], and Atlas [24]. Furthermore, domains generated by botnets, such as Zeus, Spyey, and Palevo [25], where included in the black-list.

Given the above noted datasets, the proposed botnet detection system was evaluated based on the false positive and false negative rates obtained. However, in issues such as botnet detection, there is not balance between the sizes of two classes of data (malicious and benign domains). Moreover, correct detection of the malicious group is more important than that of the benign domains. Hence, additionally, the F-score criterion is also considered. This criterion is suitable for the cases in which the two groups of data are not balanced [26]. F-score is obtained from the definitions of *Precision* and *Recall* according to Eqs. (8)-(10). TP, FP, and FN are true positive, false positive, and false negative values, respectively.

$$Precision = \frac{TP}{TP+FP}, \tag{8}$$

$$Recall = \frac{TP}{TP+FN}, \tag{9}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{10}$$

4.2. Bloom filter performance

As noted earlier, the proposed system stores a year of historical DNS traffic into the Bloom filter in order to filter long-lived domains, which are benign with high probability, as argued before. Therefore, we first eliminate popular and well-known benign domains with the help of the white-list noted earlier. After this initial filtering, about 12,800,018 domains remain which are then inserted into the bloom filter data structure. The proper size of the storage array for the Bloom filter can be calculated considering the number if inserted domains and the maximum value of 0.001% for the false positive error rate, based on Eq. (11) [17]:

$$m = -\frac{n \ln p}{(\ln 2)^2}, \tag{11}$$

where p is the given false positive probability, n is the number of elements being inserted, and m is the length of the Bloom filter.

As shown in Table 3, and given the number of inserted domains, a Bloom filter with the size of 45 MB is required and the practical false positive rate equals $7.5 * 10^{-7}\%$. Therefore, by using the Bloom filter data structure, we require a relatively small memory footprint to store the large number of historical domains.

In the second module of the proposed system, DNS domains observed would be checked as to whether they are inserted into the Bloom filter previously (i.e., were seen in historical traces), and domains not found in the historical traffic will be passed to the detection process. To calculate the performance of the Bloom filter in reducing the analyzed data, the number of requested domains is measured during the analyzed month and compared with the number of remaining domains after Bloom filter checking. The number of domains before filtering is 4,262,872 and the remaining distinct domains after comparison with Bloom filter are just 773,244 domains. Consequently, history checking reduces the test traffic by 82% (see Table 4).

Table 3. Statistical attributes of the Bloom filter.

Number of inserted records	12,900,018
Bloom filter array size	382602725 bit = 45MB
False positive error rate	7.5135E-7

4.3. Detection performance

The proposed system analyzes the DNS traffic in a short time window and aggregates the results in consecutive windows to decide about the status of each domain. In the implemented prototype, the size of time window is considered as a day and the status of each domain is determined by moving the time window and aggregating the values of the likelihood ratios in sequential windows.

The proposed mechanism needs some input parameters. First, we should select a threshold for the maximum TTL value of fast-flux domains. According to the characteristics of the current fast-flux botnets, this parameter is considered 3600 seconds [3]. Secondly, we used the traces from March 1st through 30th, 2012 (i.e., from Trace #1), as the training dataset with which the required thresholds in the SPRT algorithm were configured to provide a $FP = 0.15\%$ and $FN = 0.2\%$.

Afterwards, we examine the system performance by using Trace #1 from January 1st through 31th, 2013. Although the real traffic does not include enough fast-flux domains, the proposed mechanism detects 7 malicious domains from the available 8 domains correctly, as shown in Table 5. Hence, we obtain a detection rate of 87.5%. Furthermore, three domains are false positives of the system and the others are detected correctly.

Examples of detected domains, false positives, and false negatives are shown in Table 6. The false

Table 4. The performance of Bloom filter on one-month traffic.

Number of distinct domains before comparison with Bloom filter	4,262,872
Number of distinct remaining domains after comparison with Bloom filter	773,244
Reduction percent	82%

Table 5. Result summary of two traces.

Result summary	Trace #1	Trace #2
Number of distinct domains	180141	180159
Number of benign domains	180130	180138
Detected domains	7	17
False positive	3	3
False negatives	1	1
Detection rate (recall)	87.5%	94.44%
False positive rate	0.002%	0.001%
False negative rate	12.5%	5.55%
Precision	70%	85%
F-score	77.78%	89.47%

Table 6. Detected malicious domains, false positives, and false negatives in Trace #1.

Type	Domain name	SIP/MIP
Malicious domains	gasosvaz.ru.	0.99
	fetolbus.ru.	0.94
	fawsilom.ru.	0.91
	ecrihgep.ru.	0.73
	ikbyznod.ru.	0.72
	fyzsicat.ru.	0.71
False positives	cm.adgrx.com	0.87
	oparle.com	0.73
	api.crtinv.com	0.72
False negative	gehxehib.ru.	0.34
Inadequate info	epejanhi.ru.	0.52

positives are related to the benign domains with large pools of resolved IPs. Therefore, the proposed system mis-classifies them. On the other hand, among the existing malicious domains in the test traffic, one domain is not detected (a false negative). This domain is "gehxehib.ru." which had few requests during the analyzed month and its flux behavior was not represented during that period. In addition, the requests to this domain were distributed over several days and just one or two requests existed in each day. Hence, the proposed system was unable to detect this domain. The other malicious domain named "epejanhi.ru." receives an "inadequate information" label as the output of the system, which indicates that the SPRT method was not able to determine the state of this domain given insufficient activity in the observed period.

Afterwards, we repeated the experiment using Trace #2, while using the same SPRT threshold values as that in the previous experiment. Such an approach was essential, as there were few malicious domains in Trace #1 dataset and the evaluation metrics employed

Table 7. The detected malicious domains, false positives, and false negatives in Trace #2.

Type	Domain name	SIP/MIP
Malicious domains	brylanehome.com	0.99
	ikbyznod.ru	0.94
	carsales.com.au	0.88
	stjohnhos.co.uk	0.8
	newflirtingdates.info	0.78
False positives	cm.adgrx.com	0.87
	oparle.com	0.73
	api.crtinv.com	0.72
False negative	gehxehib.ru.	0.34
Inadequate info	epejanhi.ru.	0.52

did not properly indicate the accuracy of the system. Overall, we observed 180159 distinct domains in Trace #2. The accuracy and detection rates of the proposed system are shown in Table 5. The detection rate equals 94.4% and the false positive rate equals 0.001%. Examples of malicious domains detected and false positives are listed in Table 7. The domains which resulted in false positives and false negatives are the same as in those of Trace #1.

5. Discussions and future works

In the previous section, we evaluated the accuracy of the proposed detection scheme. One important question which one could raise is that how dependent is the proposed detection scheme on the botnet's level of activity. In order to analyze this issue, we spread DNS query/responses collected over a period of three days from the three botnet samples, over a longer period of 5 days and re-executed the experiment. Figure 4 illustrates the number of days needed to detect malicious domains before and after the time spread. Importantly, the same malicious domains are detected in both cases and the only difference is the detection time. Hence, one could conclude that when botnets reduce their activity, by generating fewer requests per day, the detection system would require more time to detect the malicious domain. This is mainly due to the fact that the proposed detection system aggregates the traffic information over sequential time windows.

Table 8 compares the proposed system with two other existing botnet detection approaches, BotGAD [16] and Fluxbuster [10]. BotGAD analyzes time windows, separately, and it calculates the similarity of host activities in independent time windows. Hence, if the activity of the botnet is spread over time, then as each time window is analyzed independently,

Table 8. Comparison between fast-flux botnet detection approaches.

Approach	**FluxBuster [10]**	**BotGAD [16]**	**Proposed technique**
Robustness against minimized activity	Low	Medium	High
History usage	Yes	No	Yes
Capability of data reduction	Low	N/A	High
Calculation overhead	High	Medium	Low

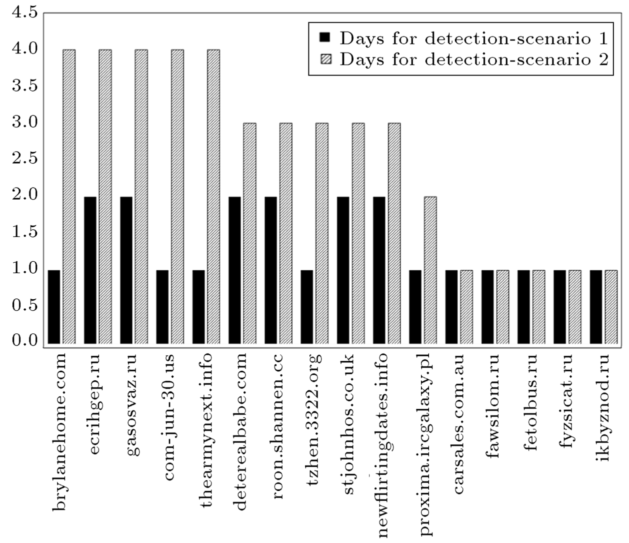


Figure 4. Comparison between the numbers of days required to detect the botnet domains, when using collected traces in 3 days, and when spreading the DNS queries over a period of 5 days.

the botnet will not be detected. Whereas in the proposed technique, information from previous time windows is also used in order to obtain more accurate results. On the other hand, Fluxbuster [10] requires a minimal number of resolved IPs for each domain in order to create a domain cluster and perform the required classification procedures. Hence, a botnet could circumvent this mechanism by sub-grouping the bots into distinct groups and avoiding the threshold required by Fluxbuster.

The other parameter of the comparison is the amount of data reduction, which in our case equals 82%. We are able to reduce the amount of analyzed DNS traffic by considering domains visited in a historical period, and still being visited today as non-malicious. This is because usually malicious domains have a short lifetime. In contrast, it seems that BotGAD performs no pre-filtering, and Fluxbuster has simple filtering rules with low effect. Finally, the proposed system has a much lower computational cost than that of the other two approaches. The proposed system only requires calculating two probability scores for each domain, while FluxBuster and BotGAD are based on machine learning and similarity-temporal correlation, respectively, which require more complicated processing. It must be noted that we were unable to compare detection accuracy results as they were dependent on the traffic used for evaluation to which we were unable to obtain access.

Nevertheless, the proposed system has its own limitations. For example, as the probabilistic functions are calculated independently in different time windows for each domain, an attacker can avoid detection by reducing the activities of the bots during each time window to a very small number of DNS queries (e.g., 3 queries, dependent on the parameters set in Section 3.3). Nevertheless, by doing so, the bots do limit their own availability. Perhaps such shortcoming could be alleviated by considering the historical states of each domain.

6. Conclusion and future works

In this paper, we propose a new system to detect fast-flux botnet domains by passive DNS monitoring. The proposed system first filters out domains seen in historical DNS traces, assuming that they are benign. We believe this assumption to be valid as legitimate domains usually have a longer lifetime, where on the other hand, botnet domains are usually short-lived. Hence, CDN domains, which are the main cause of mis-classification when looking for malicious fast-flux domains, are removed. Afterwards, a few features are calculated to help in properly categorizing the domain in question as either benign or botnet related. Our mechanism needs a small amount of data and it analyzes DNS traffic in a short and sequential time windows and calculates the final result by aggregating the output of independent time windows. The proposed system can detect malicious domains with the minimal amount of traffic by using SPRT and it provides over 94% detection rate while generating 0.001% false positive rates based on experiments using DNS traces from our campus network.

As part of the future works, we are currently considering how the proposed system could be deployed in a larger-scale network, so that it would be able to observe and detect further malicious activities. Furthermore, and in order to expand our work, we are investigating if grouping domains, resolving to the same set of IP addresses, could help in detecting domain fluxing in addition to the proposed fast-flux detection technique.

References

1. Brisco, T., *DNS Support for Load Balancing*, RFC 1794 (April 1995).
2. Cardellini, V., Colajanni, M. and Philip, S.Y. "Dynamic load balancing on web-server systems", *IEEE Internet Computing*, **3**, pp. 28-39 (1999).
3. Caglayan, A., Toothaker, M., Drapaeau, D., Burke, D. and Eaton, G. "Behavioral analysis of fast flux service networks", in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, p. 48, ACM (2009).
4. Gu, G., Perdisci, R., Zhang, J. and Lee, W. "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection", in *Proceedings of the 17th USENIX Security Symposium (Security'08)* (2008).
5. Gu, G., Porras, P., Yegneswaran, V., Fong, M. and Lee, W. "BotHunter: Detecting malware infection through ids-driven dialog correlation", in *Proceedings of the 16th USENIX Security Symposium (Security'07)* (August 2007).
6. Gu, G., Zhang, J. and Lee, W. "BotSniffer: Detecting botnet command and control channels in network traffic", in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)* (February 2008).
7. Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C. and Kirda, E. "Automatically generating models for botnet detection", in *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, Berlin, Heidelberg, pp. 232-249, Springer-Verlag (2009).
8. Holz, T., Gorecki, C., Rieck, K. and Freiling, F. "Measuring and detecting fast-flux service networks", in *Proceedings of Annual Network and Distributed System Security Symposium (NDSS'08)* (2008).
9. Perdisci, R., Corona, I., Dagon, D. and Lee, W. "Detecting malicious flux service networks through passive analysis of recursive DNS traces", in *Annual Computer Security Applications Conference (ACSAC'09)*, pp. 311-320 IEEE (2009).
10. Perdisci, R., Corona, I. and Giacinto, G. "Early detection of malicious flux networks via large-scale passive DNS traffic analysis", *IEEE Transactions on Dependable and Secure Computing*, **9**(5), pp. 714-726 (2012).
11. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W. and Feamster, N. "Building a dynamic reputation system for dns", in *Proceedings of the 19th Usenix Security Symposium* (2010).
12. Bilge, L., Kirda, E., Kruegel, C. and Balduzzi, M. "Exposure: Finding malicious domains using passive DNS analysis", in *Proceedings of Annual Network and Distributed System Security Symposium (NDSS'11)* (2011).
13. Konte, M., Feamster, N. and Jung, J. "Dynamics of online scam hosting infrastructure", *Passive and Active Network Measurement*, pp. 219-228 (2009).
14. Passerini, E., Paleari, R., Martignoni, L. and Bruschi, D. "Fluxor: Detecting and monitoring fast-flux service networks", *Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 186-206 (2008).
15. Nazario, J. and Holz, T. "As the net churns: Fast-flux botnet observations", in *3rd International Conference on Malicious and Unwanted Software (MALWARE'08)*, pp. 24-31, IEEE (2008).
16. Choi, H. and Lee, H. "Identifying botnets by capturing group activities in DNS traffic", *Computer Networks*, **56**(1), pp. 20-33 (2012).
17. Bloom, B.H. "Space/time trade-offs in hash coding with allowable errors", *Communications of the ACM*, **13**(7), pp. 422-426 (1970).
18. Lovric, M., *International Encyclopedia of Statistical Science*, Springer London (2011).
19. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. and Euler, T. "Yale: Rapid prototyping for complex data mining tasks", in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 935-940, ACM, (2006).
20. Tan P.N., Steinbach M. and Kumar V., *Introduction to Data Mining*, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005).
21. "Alexa top sites" (2014). http://www.alexa.com/topsites.
22. "Malware domain list" (2014). http://www.malwaredomainlist.com/.
23. "Dns-bh - malware domain blocklist" (2014). http://www.malwaredomains.com/.
24. "Atlas" (2014). http://atlas.arbor.net/summary/fastflux/.
25. "The Swiss security blog" (2014). https://abuse.ch.
26. Pang-Ning, T., Steinbach, M., Kumar, V. et al., *Introduction to Data Mining*, in Library of Congress, p. 74 (2006).

Biographies

Elahe Soltanaghaei received her BS degree in Computer Engineering, with honors, from Amirkabir University of Technology, Tehran, Iran in 2011 and her MS degree in Computer Engineering from Sharif University of Technology, Tehran, Iran, in 2013. She is currently a researcher with the Compuco International Co. (International Banking, Finance, and IT Consulting-

Iran/Switzerland). Her main research interests are computer networks, network security, and information security.

Mehdi Kharrazi received his BE degree in Electrical Engineering from the City College of New York and his MS and PhD degrees in Electrical Engineering from the Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, New York, in 2002 and 2006, respectively. He is currently an Assistant Professor with the Department of Computer Engineering, Sharif University of Technology, Iran. His current research interests include network and multimedia security.