2014

ADAPT: an anonymous, distributed, and active probing-based technique for detecting malicious fast-flux domains

Tsolmon Otgonbold
Iowa State University

ADAPT: An anonymous, distributed, and active probing-based technique for detecting malicious fast-flux domains

by

Tsolmon Otgonbold

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Information Assurance

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Yong Guan, and my committee members, Daji Qiao, and Ying Cai, for their guidance and support throughout the course of this research.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.

Finally, thanks to my family for their encouragement and to my wife for her hours of patience, respect and love.

ABSTRACT

The fast-fluxing has been used by attackers to increase the availability of malicious domains and the robustness against detection systems. Since 2008, researchers have proposed a number of methods to detect malicious fast-flux domains, however they have some common drawbacks in the system design, which are as follows: no anonymity, partial view on the domain, and unable to detect before an attack takes place. Therefore, to overcome these drawbacks, we propose a new technique called ADAPT, which enables a detection system to collect DNS information of a domain anonymously all around the globe in short period of time with less resource using Tor network.

In this thesis, we have developed a prototype of ADAPT, which takes its input from domain zone files to detect in-the-wild malicious fast-flux domains. We defined a flux score formula to propose 10 new detection features. The prototype of ADAPT has scanned over 550,000 .net domains, and extracted 20 distinct features for each of the domains.

By analyzing the obtained DNS dataset, we observed several new findings and confirmed some new trends reported in the previous researches. Moreover, our experimental result showed that the prototype of ADAPT has a potential to outperform the existing detection systems, with a few modifications and updates in the detection process.

CHAPTER 1. INTRODUCTION

1.1. Research Motivation

Along with the Internet development, cyber criminals find new ways to cover the traces of their illegal activities. In the past years, they have developed a new evasion technique called fast-flux, which shares similar characteristics with DNS based load balancing techniques such as content-delivery network (CDN) and Round-robin DNS (RRDNS). These techniques provide a high degree of availability, scalability, and performance to high volume Internet services. The RRDNS distributes clients' requests to different physical servers by shuffling the IP addresses of a domain to support load balancing. The CDN does the same, however, since it is a network of a relatively large number of nodes scattered across multiple locations around the world, it takes the client's geographic location into account, and returns the IP address of the nearest available node to the client to speed up the service. Similarly, the fast-flux uses the same concept to constantly change the physical host that a domain corresponds to, which enables cyber criminals to prevent from being detected. One of the illegal activities in the Internet is a fake online pharmacy, which is known to be selling counterfeit or substandard medications, and engaged in identity theft [1]. According to the report from Fortinet Global Security Research Team, fake Canadian online pharmacies have been using the fast-flux technique to avoid easy take down [2].

Fast-flux networks (FFNs) differ from legitimate CDNs by their nodes. The nodes in a CDN are professionally administrated machines, whereas the nodes in a FFN, also known as flux-agents, are malware-infected machines. The flux-agents are often found to be a part of a botnet, and remotely controlled by the botmaster. Botnets that use the fast-flux technique, can obtain an extra layer of protection by using flux-agents as proxies that relay user request to backend servers, also known as motherships, because the frequent and fast change of flux-agents in the FFN makes it more difficult to track down criminal activities and shut down their operation. Recently, security researchers reported that a new variant of “ZeuS Gameover” botnet employs the fast-flux technique to hide its command and control (C&C) servers [3].

Since 2008, the increasing usage of fast-flux technique by cyber-criminals has drawn great attention from both academic and industrial fields. The most common approach detecting fast-flux domains is to count distinct IP addresses that resolve to a suspected domain by analyzing DNS traffic. Also, the number of distinct Autonomous System Numbers (ASN), network addresses, organization names, and country codes from the collected IP addresses can be counted to increase the detection rate [4]–[11]. As the value for these numbers increase, the probability of being fast-flux domain increases. However, all these proposed detection techniques claim to have high detection rates and low false positives, they have two major disadvantages in their system architecture.

First, the input to the system is either obtained from DNS traffic traces or spam filters. They both tend to increase the detection delay, which allows malicious users to move to the other corner. For the systems that take a DNS traffic trace as an input, the coverage depends on where and how many different DNS servers the system collects the DNS traffic from. In order to detect a malicious fast-flux domain, at least one user in the network need to make a request to the malicious domain, which is the main cause of the delay. Similarly, for the systems that take input from spam filters, at least one incident must have been occurred in order to start investigating the domain.

Second, the view on a suspected domain is limited for all the existing systems. As mentioned before, CDNs return different IP address for a domain depending on the user location, FFNs have the potential to do the same to evade the detection. To increase the view, existing systems must deploy the systems at multiple locations around the globe. First of all, it is costly. Second, it requires a lot of effort from multiple parties.

In this work, we propose an Anonymous, Distributed, and Active Probing-based Technique (ADAPT) to detect malicious fast-flux domains. ADAPT overcomes all the disadvantages of the existing systems mentioned above. ADAPT takes its input from domain zone files, which contain the change of the DNS information of a domain. Since, the fast-flux requires frequent change in DNS information, domain zone files can provide delay free input to the detection system. Also, ADAPT uses the Tor network to have a worldwide view on a suspected domain, without a need of deploying the system in multiple locations across the globe. Another advantage of using Tor is it provides anonymity to the detection system, hence a malicious domain has to consider our active probing clients as one of its victims, thus providing its true identity. Then, using a new formula of flux score that we propose, ADAPT calculates a set of flux scores based on the IP addresses, ASN, network addresses, organizations and country codes, that each of the suspected domains resolved to, in order to detect malicious fast-flux domains.

We implemented a prototype system of our proposed technique ADAPT, and experimented with .net top level domain's zone file. Overall, our client actively probed more than 550'000 domains, which changed their DNS information in one day. Analyzing the collected data, we observed several new findings, and our experimental result showed that the proposed technique is capable of detecting malicious fast-flux domains.

1.2. Thesis Scope

The scope of thesis is to implement a prototype of the proposed technique and analyze the collected data for detecting malicious fast-flux domains. However, it does not include fully working detection system or analysis of all the available detection features.

1.3. Thesis Organization

The rest of this thesis organized as follows: Chapter 2 gives the technical background information of the fast-fluxing. Chapter 3 covers the literature survey on existing researches on detecting malicious fast-flux domains. Chapter 4 defines the formal statement of the problem being solved along with the structure of the data set and previously proposed detection features. Chapter 5 introduces the high level design of ADAPT, and its functional and implementation details. Chapter 6 details the result of analysis on the collected data obtained by using the prototype of ADAPT, and concludes the findings observed from the analysis. Chapter 7 discusses the limitations of this work and proposes solutions to the limitations. Chapter 8 summarizes the thesis and presents potential future improvements.

CHAPTER 2. PRELIMINARY: THE DNS SYSTEM, ITS DEPLOYMENT VARIATIONS, AND FAST-FLUXING

Fast-fluxing can be implemented on top of Domain Name System (DNS). Therefore, it is better to have a look at the underlying technology and its deployment variations, which will help us to understand the fast-flux technique in the end of this chapter.

2.1. Overview of DNS

DNS [12] is one of the fundamental component of the Internet and is essential to looking up resources in the Internet. DNS basically translates a domain name to its corresponding IP addresses and vice versa. Also, it identifies related resource information such as authoritative name server, SMTP email server, etc.

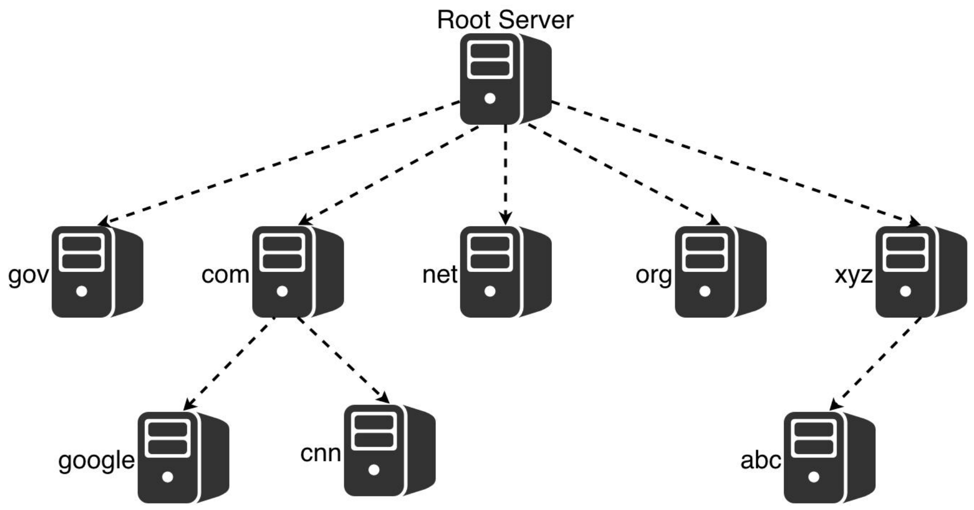

Figure 1. *DNS hierarchical tree structure*

DNS is a hierarchically formed tree structured distributed system as shown in figure 1, where every node is responsible for storing resource records (RR) of its sub nodes in a file called zone file. For example, the root nodes are responsible for Top Level Domain (TLD) nodes such as .com, .net, .org, which are responsible for Second Level Domain (SLD) nodes such as cnn.com, Wikipedia.org, and so on. RR is the basic data element in the DNS, which consists of a name of the parent node, a record type (A, MX, etc.), a class code, an expiration time limit, and some-type specific data.

Also, the DNS can be seen as a server-client system, where the servers hold RRs, and the clients make a request to the server to obtain a necessary RR. The request can be either recursive or iterative. A recursive query demands the final answer to be returned in the response, and a server which has received a recursive query should handle all the other queries to obtain the final answer. For an iterative query, the answer in the response does not have to be final. It can redirect the client to another server which should have the RR the client requested in the first place, and the client should make another request to that server. This process is continued until the client obtains the final answer.

The server holding RR for a domain is called authoritative name server (AuthNS) of that domain. Usually more than one AuthNS store same RRs, in order to achieve fault tolerance. On the other hand, the clients are called DNS resolvers, and they initiate DNS queries. A user

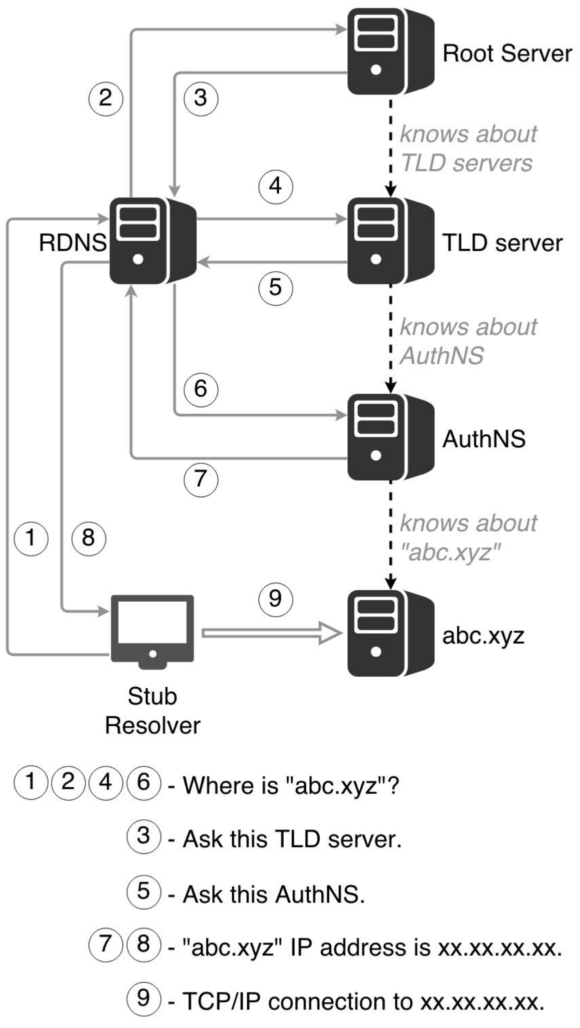

Figure 2. *DNS – Domain resolution process*

machine is called stub resolver in the DNS, and it sends a recursive DNS query to a recursive DNS resolver (RDNS), which then resolves the request by sending iterative DNS queries to name servers starting from a root server to an AuthNS. RDNS are provided by Internet Service Providers (ISPs) or other organizations to increase the performance of stub resolvers.

In the figure 2, a client machine sends a request to an RDNS to find out the IP address of “abc.xyz”. To resolve the request, it sends a query to one of the well-known root servers, which then replies with the IP address of a responsible TLD server. RDNS sends the same query to the TLD server, which replies with the IP address of AuthNS of “abc.xyz”. Once again, RDNS sends the same query to the AuthNS, which sends back the IP address of “abc.xyz”. Finally, RDNS replies back to the client machine with the resolved IP address of the requested domain name. So, now the client machine can connect to “abc.xyz” through TCP/IP protocol.

2.2. DNS Deployment Variations

In order to improve the performance of the Internet services, some functional improvements can be made in the DNS. As mentioned before, a RR has an expiration time limit, which is known as Time To Live (TTL). To minimize the number of queries in the DNS, name servers cache the RR for a period of time of its TTL value after the DNS response has received.

Web sites with high traffic use Round Robin DNS (RRDNS) and Content Delivery Network (CDN) to handle their traffic load. These techniques distribute the traffic to different physical servers by resolving the domain name to different IP addresses. Depending on the configuration, the RRDNS returns either a same set of IP addresses in a different order, or a different single IP address in a Round Robin fashion, each time a client sends a DNS query.

CDN is a large distributed system of servers that deliver the content to a client based on the geographic location of the client. Similar to RRDNS, CDNs use DNS to redirect a client to a server which is the nearest to the client [13]. Both RRDNS and CDN use low TTL value to increase the efficiency of traffic distribution and the responsiveness to changes in link characteristics. However, using a low TTL value increases the load on the DNS by minimizing the duration of RR caching [14].

2.3. Fast Fluxing

A domain that uses a RRDNS or a CDN, is resolved into multiple IP addresses. As long as one of these IP addresses responds, the entire service is online. Fast-fluxing employs the same idea in a malicious way. The main difference between CDN and fast-flux is that a fast-flux domain is hosted on a large number of machines, also known as flux-agents, that can be taken down or go off-line anytime, whereas a domain in a CDN is hosted on a large pool of professionally administrated servers, that rarely go off-line. Therefore, the total number of distinct IP addresses that a fast-flux domain is resolved to, tend to be larger than the total number of distinct IP addresses that a domain in a CDN is resolved to. Usually, an attacker uses compromised machines as flux-agents, to build a fast-flux network (FFN), which is cheaper, more resilient, and more robust against mitigation. A FFN can be used for a number of malicious activities in the Internet such as phishing, spamming, illegal content hosting, etc.

Fast-flux domains are categorized as single-flux and double-flux domains. A single-flux domain changes only the IP addresses of its host servers, whereas a double-flux domain changes the IP addresses of its host servers as well as the IP addresses of its AuthNSs. In order to make a fast-flux domain to be more resistant to discovery and mitigation, a second layer called blind proxy redirection can be added, where flux-agents act as proxies to route an incoming request to one of the main servers, also known as motherships. This second layer prevents the fast-flux nodes from being tracked down, and helps the attacker to use less resource, because the attacker does not have to maintain every single flux-agent as a server hosting malicious content. Instead, the attacker keeps one or two motherships online and regularly update the RRs of the fast-flux domain in the DNS to point to different set of blind proxies. When a request comes to a blind proxy, it passes the request to a mothership, and when a response comes back from the mothership, the blind proxy passes it back to the client who made the initial request.

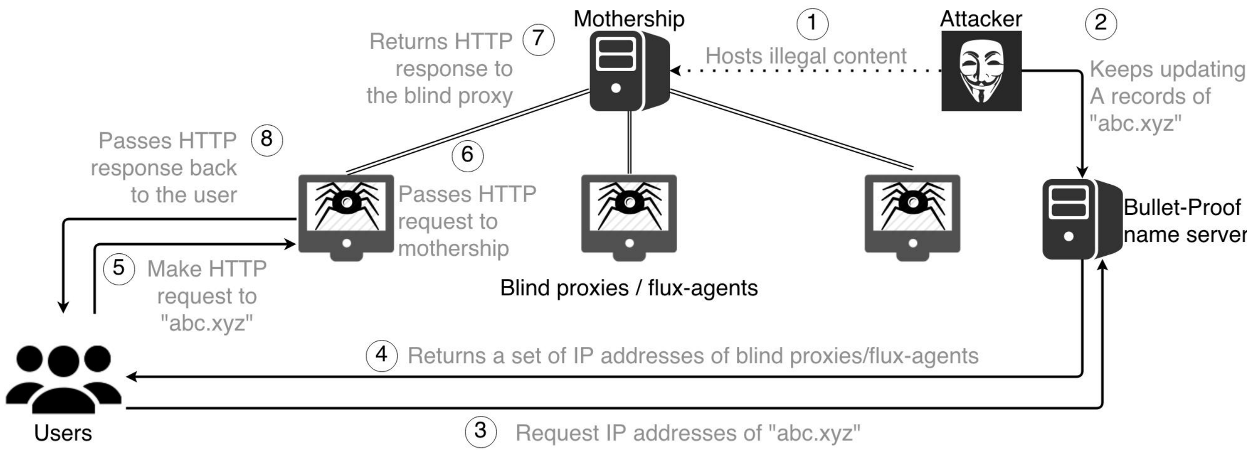
Mothership
1
Attacker
2
Returns HTTP response to the blind proxy
7
Hosts illegal content
Keeps updating A records of "abc.xyz"
Passes HTTP response back to the user
8
6
Passes HTTP request to mothership
Bullet-Proof name server
5
Make HTTP request to "abc.xyz"
Blind proxies / flux-agents
4
Returns a set of IP addresses of blind proxies/flux-agents
Users
3
Request IP addresses of "abc.xyz"

Figure 3. *Single flux*

In figure 3, a single-flux network is illustrated, where an attacker updates A records of the domain “abc.xyz” in a bullet-proof DNS server pointing to a set of blind proxies, which pass HTTP requests from users to the mothership. When a client wants to access the domain “abc.xyz”, the bullet-proof DNS server returns a current set of IP addresses of blind proxies. Then using one of the returned IP addresses, the client connects to the malicious host server through one of the proxies.

A double-flux network is depicted in figure 4. A mothership hosts both DNS service and web service, and an attacker updates NS records of the domain “abc.xyz” on a legitimate DNS server. A client sends a query to the legitimate DNS server to resolve “abc.xyz”, then the DNS server replies back with a set of IP addresses of current proxies. The client again sends the DNS query to one of the returned proxies, which pass the query to the mothership. The mothership replies with IP addresses of a set of blind proxies from the pool. The client then makes an HTTP request to the mothership via one of the proxies later returned.

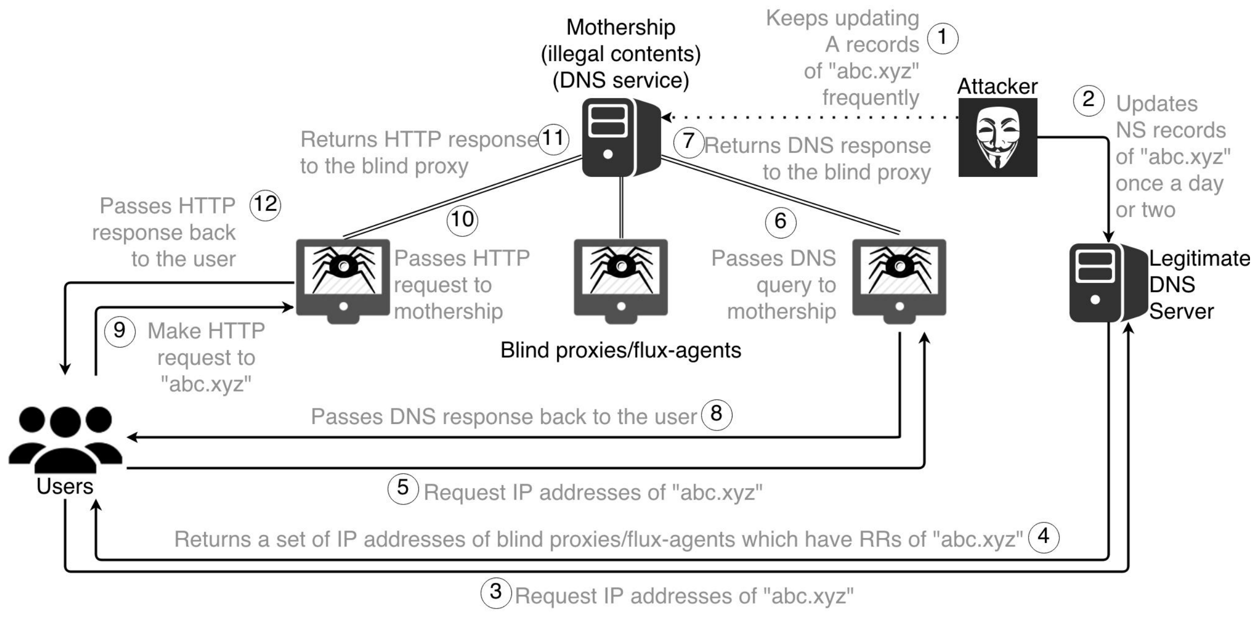
Mothership
(illegal contents)
(DNS service)
Keeps updating A records of "abc.xyz" frequently 1
Attacker
2 Updates NS records of "abc.xyz" once a day or two
Returns HTTP response to the blind proxy 11
7 Returns DNS response to the blind proxy
Passes HTTP response back to the user 12
10
6
Passes HTTP request to mothership
Passes DNS query to mothership
Legitimate DNS Server
9 Make HTTP request to "abc.xyz"
Blind proxies/flux-agents
Passes DNS response back to the user 8
Users
5 Request IP addresses of "abc.xyz"
Returns a set of IP addresses of blind proxies/flux-agents which have RRs of "abc.xyz" 4
3 Request IP addresses of "abc.xyz"

Figure 4. *Double flux*

In conclusion, fast-fluxing gives an attacker three major advantages. First, it offers simplicity, where the attacker can use few powerful back-end servers as motherships. Second, an extra layer of protection against tracking and discovery is offered by blind proxies. Third, the operational life spans of motherships are extended by the extra layer of protection.

CHAPTER 3. LITERATURE SURVEY

Since 2008, many researchers have studied fast-flux domains, and proposed several different detection techniques. They differ by the data they use, and the methods used to obtain the data. Therefore, we have categorized them as DNS based, non DNS, and hybrid techniques.

3.1. DNS Based Detection Techniques

A number of researchers have shown that the DNS traffic analysis is an effective approach to detect fast-flux domains. Most of the existing works have used an active probing method to obtain the DNS traffic. Some have chosen a passive monitoring approach to avoid interaction with suspected domains. And, a few works have combined the both methods to increase the detection rate.

3.1.1. Active probing method

In [4], the authors presented the first empirical study of fast-flux networks, and measured the extent of FFN with help of temporal-based characteristics and spatial features. A general metric named Flux-Score was proposed as a decision maker, which is derived by considering the number of unique A records in overall DNS lookups, the number of NS records in a single DNS lookup, and the number of unique ASNs for overall A records for a domain. If the Flux-Score is less than or equal zero, the domain is considered as benign, otherwise, the domain is classified as fast-flux domain. However, the Flux-Score was claimed to be very accurate in [4], the detection delay becomes very long for benign domains, due to temporal-based characteristics used in the detection, which usually requires a period of time of at least TTL seconds.

In [6], the authors proposed a bit different formula of the Flux-Score with some adjustment numbers like TTL value is less than 3600 seconds, the number of IP addresses returned in the DNS response is greater than or equal to 5, etc. However, the authors claims that the system detected many FFNs, it might have high false negatives due to those static adjustment numbers in the proposed formula. If not, once the formula is available to public, fast-flux domain owners can change the characteristics easily to avoid from being detected.

To decrease the long detection delay in [4], the authors in [15] developed a new FFN detection system, FluXOR, which collects suspicious domains from spam emails in honeypots, and monitors them by querying non-AuthNSs and WHOIS servers over a period of three hours. The collected data about a suspected domain is fed to a naïve Bayesian classifier to classify as fast-flux or benign. It uses nine different features for classification, and they are grouped into three categories: the domain of the suspected host name, the degree of availability, and the heterogeneity of the hosts in the target network. Although, FluXOR used a time threshold of 3 hours on TTL value to reduce the detection time, it is still relatively long for detecting a single malicious fast-flux domain.

In [16], the authors proposed a delay-free detection system, Spatial Snapshot Fast-flux Detection system (SSFD) for identifying FFN in real-time. The main feature of this work is it takes geographic locations of resolved IP addresses of a suspected domain into an account, which is represented in two spatial measures: spatial distribution estimation, and spatial service relationship evaluation. However, SSFD presents a high detection rate and low false positive, it suffers from unavailable location data of an IP address, which limits the effectiveness of this scheme in detecting FFN.

3.1.2. Passive monitoring method

In [5], the authors proposed a novel passive approach for detecting malicious FFNs. The proposed system collects RDNS traffic traces from multiple large networks, which offers the system an anonymity and an ability to detect in-the-wild malicious domains. The detection system groups all the domains according to the similarities in their resolved IP sets, because a FFN uses a large number of domain names that all point to same flux-agents to evade domain blacklists. For each cluster of domains, passive and active, total 12 features are extracted over a moving window of 1 day. After measuring the features, C4.5 decision-tree classifier is employed to classify domains as benign or FFN. However, this system has a high detection rate and some other advantages over previous works, there are two major drawbacks. First, it has partial visibility on domains, because it collects its input from a limited number of RDNS. Second, detection delay is longer than all other works, because it needs to monitor a domain for 5 days to detect its uptime.

In [9], the authors employed both real-time classification and long-term monitoring approaches. Similar to [5], the proposed system takes its input from multiple RDNS traffic traces. For real-time classification, it extracts six features from DNS responses in the DNS traffic traces. During 48 hours long long-term monitoring, if the numbers of IP addresses or networks associated with a suspicious domain reaches a certain threshold, it is determined to be fast-flux, otherwise, it is put into the whitelist. This work has shown that the number of distinct ASNs for all A records offers the highest information gain. The detection statistics of this system shows that it detected 2551 fast-flux domains, and generated 3104 false positives over 180 days long experiment, which is a relatively high value for false positives.

3.1.3. Combination of both methods

To further decrease the detection time, the authors of [17] has proposed the Fast-Flux Monitor (FFM), a real-time detection of fast-flux domains. FFM uses both active and passive DNS monitoring to achieve real-time detection. Active monitoring collects three features for a domain: TTL, FF Activity Index, which is a short term statistic of the change in distinct A records over a moving window of 10 minutes, and Footprint Index, which is a number of unique ASNs for all A records of a suspected domain. Passive monitoring replicates the active monitoring, but it only considers DNS responses from AuthNS. Furthermore, FFM uses analytic sensors to calculate the association between fast-flux networks. Similar to FluXOR, it uses Bayesian classifier to classify suspected domains as benign or fast-flux. However, the authors of FFM claim that the detection is in real-time, it requires a domain to be monitored over several moving windows of ten minutes to increase the detection accuracy.

3.2. Non DNS Detection Techniques

Another real-time detection system is proposed in [18]. The authors of this work took different approach to detect FFN by looking at the HTTP request resolution time, not DNS traffic. Also, it makes three assumptions on hosts in FFN: they pass all the valid HTTP request to their motherships, they have low bandwidth, and their performance is low. However, these assumptions do not always have to be true. And benign web servers can have same characteristics. For instance, a proxy server sends all the valid HTTP traffic to the main server. Thus, it leads to high false negatives and false positives.

In [19], the authors have developed a software, Fast-Flux Watch (FF-Watch), to detect any possible flux-agents in a stub network by correlating incoming TCP connection requests to flux-agents with outgoing TCP connection requests from the same agents. The FF-Watch provides a protection to a stub network by installing it on a leaf router that connects the network to the Internet. To affect the performance of a FFN, the FF-Watch need to be installed on many leaf routers, which is hard to achieve.

3.3. Hybrid Detection Techniques

In [8], the authors experimented a number of sets of features to detect FFN, where the sets are comprised of timing-based, network-based, spatial-based, domain name-based, and DNS answer-based. The experiment employed C4.5 decision tree classifier to classify total 476 fast-flux domains and 1853 benign domains. The experimental result presented that spatial-based features performed best, followed by network-based, DNS answer-based, domain name-based, and timing-based features. When all the sets of features are jointly used together, the detection accuracy reached 98.9%, and the predictions become insensitive to the timing-based and domain name-based feature sets.

In [10], the authors have combined two distinct sets of features from [18] and [20] to complement drawbacks in the two approaches. The result shows the combination performs better than each of the individual works. However, the authors mention that FFN still could evade this proposed detection method theoretically.

Similarly, in the Genetic-based Real-time Fast-flux Service Networks Detection (GRADE) [7], the authors employed same ideas from [18], [20]. However, it proposed a new detection feature, which is the entropy of domain names of preceding nodes (E-DPNs) of all A records for a domain. The preceding nodes can be found by using “traceroute” command. If the E-DPNs of a domain is high, the domain is categorized as fast-flux domain. Although, the detection rate is high and delay is short, the attackers can evade GRADE by either providing a single A record in the DNS response or providing a set of A records that are geographically close to each other.

3.4. Summary

As a result of this literature survey, we summarized the common drawbacks in the existing systems as follows:

1. All of the existing systems can be considered as reactive, because they take their inputs from user generated databases such as spam trap, DNS traffic, and TCP traffic, in which malicious domains get registered only after the attack has been launched.
2. Most of the existing systems do not consider that FFNs can detect the detection systems and provide false identity. Only [9] can be counted as an anonymous system, however, it suffers from a partial view on suspected domains due to its passive monitoring approach.
3. A well-developed FFN could act like a benign CDN by returning a set of A records that are located geographically close to each other. Therefore, existing detection systems that use features related to the IP address could misclassify those well-developed FFNs as benign CDNs, because they collect the DNS traffic from a limited number of locations.
4. Finally, [21] has reported that FFNs have started using a higher TTL value to avoid being detected. This means FFN developers can change the characteristics of fast-fluxing to evade detection systems, even if the change affects the performance of the FFN. Therefore, upcoming detection systems need to use FFN characteristics that are not easy to change.

CHAPTER 4. PROBLEM DEFINITION

The formal problem statement can be formulated as follows: design and build a DNS data collection system to detect malicious fast-flux domains. Since malicious domains tend to have a shorter life span [22], an ability to detect malicious fast-flux domains in their infancy is desired. Also, DNS servers may return different RRs of a domain to a client depending on the client's location. Therefore, for more accurate detection, it is important to collect all the IP addresses of the RRs of a domain, which can be fulfilled by sending a set of spatial and temporal DNS queries, i.e., a set of DNS queries sent from different locations to different name servers for a several times.

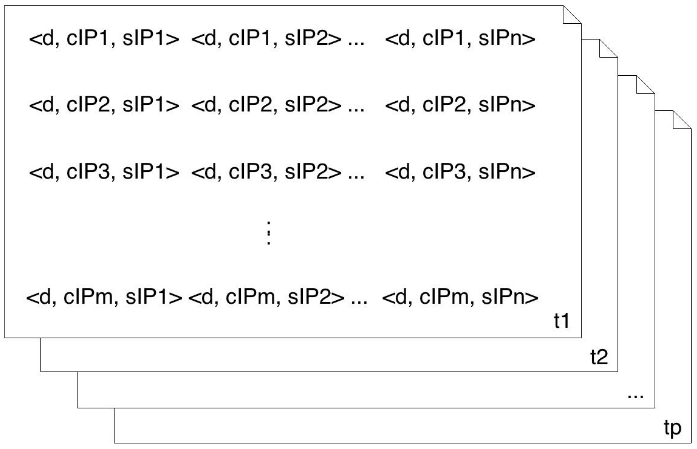

Figure 5. *Spatial and temporal DNS query set*

In figure 5, a spatial and temporal DNS query set is illustrated for a domain d, where the DNS queries are sent for p times from every m places to every n name servers, and a DNS query in the set is seen as a vector of the domain name, an IP address of a client, and an IP address of a name server, v_{query}=<d, cIP, sIP>.

A single DNS query results a set of DNS responses from a name server. and similarly a DNS response can be expressed as a vector, $v_{response}$=<TTL, t, type, IP>, where:

- TTL is an expiration time of the DNS information, which is expressed in seconds,
- t is a time stamp indicating when the response is received,
- type indicates whether the IP address is of a host server or a name server of the domain,
- IP is an IP address of a resource of the domain.

For example, v_{query}=<“iastate.edu”, “69.5.147.140”, “8.8.8.8”> can result following DNS responses:

$v_{response}$=<21599, “Tue Oct 28 13:35:45 CDT 2014”, “host”, “129.186.1.99”>,

$v_{response}$=<21599, “Tue Oct 28 13:35:45 CDT 2014”, “name server”, “129.186.6.249”>,

$v_{response}$=<21599, “Tue Oct 28 13:35:45 CDT 2014”, “name server”, “129.186.88.99”>.

Once, the DNS responses for the set of spatial and temporal DNS queries are obtained, all the possible features should be extracted from the collected data set for the domain classification as benign or malicious fast-flux domain. Table 1 shows all the features proposed by researchers who studied DNS traffic to detect malicious domains. However, some of them require extra information.

- Feature 1 may be obtained by querying WHOIS databases, however not all the domain information is available in a single WHOIS database.
- Features 2, 3, 4, 5 and 19 can be extracted directly from the spatial and temporal DNS traffic.
- Features 6 – 13 can be calculated after mapping a IP to ASN, network, organization name, and country code using some online databases, such as [23].

Table 1. All the available features proposed in the previous works.

#	Feature	Explanation
1	WHOIS information	Domain created date, updated date, expire date
2	TTL of records related to host servers	Calculating statistical value of TTL of A and AAAA records of host servers (ANSWER)
3	TTL of records related to name server	Calculating statistical value of TTL of A and AAAA records of name servers (ADDITIONAL RECORDS)
4	Distinct IP addresses of host server	Counting distinct IP addresses (v4 and v6) of host servers
5	Distinct IP addresses of name server	Counting distinct IP addresses (v4 and v6) of name servers
6	Distinct ASNs of host server	Counting distinct ASNs of IP addresses of host servers
7	Distinct ASNs of name server	Counting distinct ASNs of IP addresses of name servers
8	Distinct networks of host server	Counting distinct networks of IP addresses of host servers
9	Distinct networks of name server	Counting distinct networks of IP addresses of name servers
10	Distinct countries of host server	Counting distinct country codes of IP addresses of host servers
11	Distinct countries of name server	Counting distinct country codes of IP addresses of name servers
12	Distinct organization of host server	Counting distinct organization names of IP addresses of host servers
13	Distinct organization of name server	Counting distinct organization names of IP addresses of name servers
14	Domain registrar	Calculating reputation score of domain registrar
15	NS reputation score	Calculating reputation score of name server
16	Guilt by association score	Calculating reputation score of IP address
17	Service distance	Counting number of hops from client to host
18	Entropy of domain names of preceding nodes	Calculating entropy of domain names of nodes between front-end server and back-end server
19	DNS query failure rate	Counting NXDOMAIN responses
20	Similarity of scam pages	Calculating similarity of web pages using String kernel
21	Service banner	Version and configuration information of web services

- Feature 14 needs WHOIS and historical data. Map a domain name to its registrar, then count total numbers of malicious domains registered by each and every registrar.
- Features 15 and 16 require some amount of historical data. For every distinct IP address, a total number of malicious domains that map to the IP address has to be counted.
- Feature 17 counts the number of hops between a client and a host server, or a host server and a name server that is queried by a client to obtain DNS information of a domain. The number of hops can be counted with a traceroute command which returns the nodes between two devices in a network.
- Feature 18 analyzes the reverse domain names of nodes in between a front-end server and a back-end server of a domain, which can be obtained with help of a set of traceroute commands and reverse DNS lookup queries.
- Features 20 requires the content on the domain to be downloaded, which can be done by sending HTTP GET request to a host server of a domain.
- Feature 21 requires the content of each of the nodes in between a front-end server and a back-end server of a domain, which can be obtained by a set of traceroute commands and HTTP GET requests.

Finally, the new detection technique should resolve the common drawbacks of the existing systems that are summarized in Section 3.4.

CHAPTER 5. PROPOSED APPROACH

5.1. High Level Description of System Architecture

In this thesis, we are proposing an Anonymous, Distributed, Active Probing Technique (ADAPT) to collect DNS data for detecting malicious fast-flux domains in the Internet. And, we have implemented a prototype of ADAPT, which consists of a server and a number of clients. We chose 3 tier client-server architecture to achieve the goal of scalability.

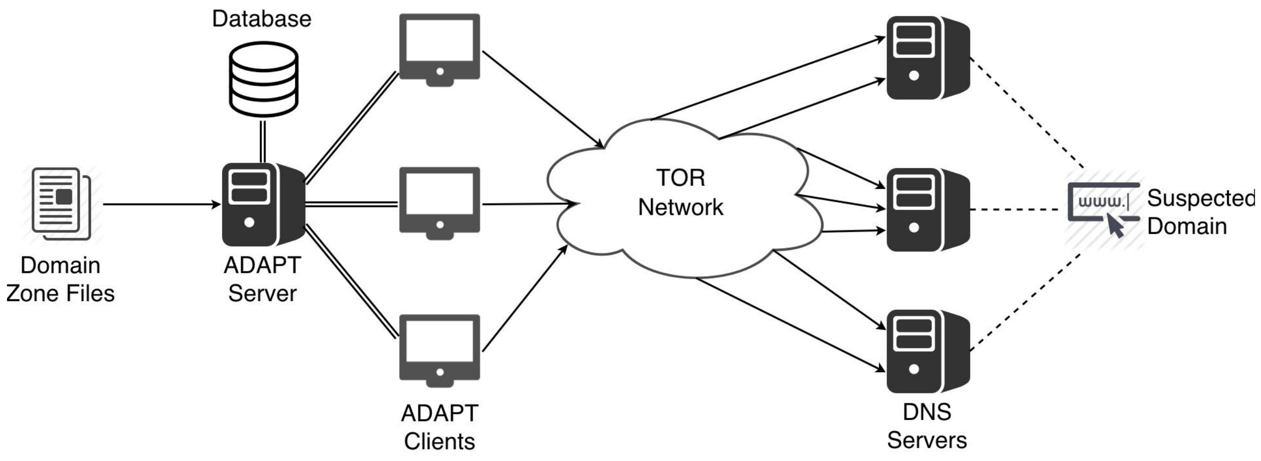

Figure 6. *ADAPT - System architecture*

In order to detect in-the-wild malicious domains, the ADAPT takes its input from domain zone files, because several researchers have shown that domain zone files can be used to detect malicious domains in their infancy [24]–[26]. The ADAPT server extracts all the domains that have changed their records in two sequential zone files, and considers them as suspicious domains, because the main characteristic of the fast-fluxing is to update the DNS information of a domain frequently. Therefore, the ADAPT system has a high potential to detect a malicious fast-flux domain even before any malicious activity takes place involving the domain.

Detection systems that need to interact with the suspected domains in one way or another, should employ an anonymizer to avoid direct interactions. We have employed Tor network as an anonymizer, because it offers an anonymity as well as a worldwide distributed network. Tor is a free, worldwide network of thousands of volunteer relays to conceal user's location and identity. In the Tor network, the user traffic is sent through a randomly constructed chain of relays. The relay in the end of the chain is called exit node, which may be in any region of the world. Thus, it enables ADAPT to collect spatial and temporal DNS traffic from all over the world just by sending DNS queries from a single point through different chains in the Tor network.

Out of 21 features listed in the table 1, chapter 4, we have employed only the features 4 to 13 in the prototype of ADAPT due to time limitation. To extract these features, ADAPT maps all the IP addresses to their ASNs, network addresses, organization names, and country codes, with the help of an online database [23], which can be accessed by sending a specially crafted reverse DNS query to a recursive DNS server. However, the features 4 to 13 have a direct relationship with the number of DNS queries that have been sent, i.e., their values tend to increase as the number of queries increases, which makes them inefficient to use directly in the malicious fast-flux domain detection. Therefore, to normalize the values of these features, we propose a new Flux-Score formula to indicate the average number of new values discovered in a single DNS query, which is defined as the following:

Given a set of DNS responses of n number of DNS queries, $A_1 \ldots A_{n-1}, A_n$

Flux-Score, $FS = \frac{\sum_{i=1..n} FS_i}{n}$ where $FS_i = \frac{|A_i - \cup_{k=1..i} A_k|}{\frac{\sum_{k=1..i} |A_k|}{i}}$ is the number of new values discovered in the i -th query divided by the average number of values returned for a query.

Table 2. Features extracted by the prototype of ADAPT.

#	Abbreviation	Explanation
1	FSip-hs	Flux-Score of IP addresses of host servers
2	FSip-ns	Flux-Score of IP addresses of name servers
3	FSasn-hs	Flux-Score of ASNs of host servers
4	FSasn-ns	Flux-Score of ASNs of name servers
5	FSnet-hs	Flux-Score of network addresses of host servers
6	FSnet-ns	Flux-Score of network addresses of name servers
7	FScc-hs	Flux-Score of country code of host servers
8	FScc-ns	Flux-Score of country code of name servers
9	FSorg-hs	Flux-Score of organization name of host servers
10	FSorg-ns	Flux-Score of organization name of name servers
11	Dip-hs	Distinct IP addresses of host server
12	Dip-ns	Distinct IP addresses of name server
13	Dasn-hs	Distinct ASNs of host server
14	Dasn-ns	Distinct ASNs of name server
15	Dnet-hs	Distinct networks of host server
16	Dnet-ns	Distinct networks of name server
17	Dcc-hs	Distinct countries of host server
18	Dcc-ns	Distinct countries of name server
19	Dorg-hs	Distinct organization of host server
20	Dorg-ns	Distinct organization of name server

By applying this Flux-Score formula on the existing features 4 to 13, we have obtained 10 new features. Table 2 shows all the features that the prototype of ADAPT extracts for a domain from the obtained DNS traffic, in order to classify the domain as malicious fast-flux domain.

5.2. Functional Details

In ADAPT, the messages passed between a client and a server are of two types: client job and client job response. Client job consists of a suspected domain name, a probe intensity level, and id for internal use. A client job response contains a client job, a set of name server IP addresses that returned a NXDOMAIN response, a set of name server IP addresses that failed to resolve the domain, and a set of DNS responses. A DNS response consists of an IP address of the exit node, an IP address of a name server queried, a resolved IP address, a boolean value indicating if the resolved IP is for name server, a TTL value, and a time-stamp of the response. Figure 6 shows the class diagrams of client job, client job response and DNS response.

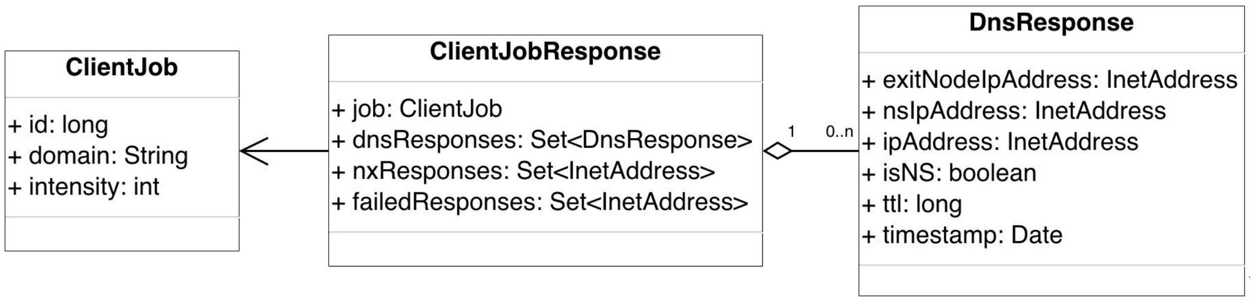

Figure 7. *Class Diagram – Messages passed between client and server*

The clients fetch client jobs from the server, and return the client job responses as soon as they have collected enough data for the jobs. In the following sections, we describe the functional details of the ADAPT server and the ADAPT client.

5.2.1. ADAPT server

The ADAPT server is responsible for a several activities, such as extracting suspicious domains from zone files, creating and distributing client jobs, receiving client job responses, and classifying the suspected domains. For classification, total 20 features are calculated for a domain, however if required information is missing, the server may query 3rd party online databases to obtain necessary information, or create another client job for the domain.

Client jobs are created from the differences found in two consequent zone files of a domain. If a domain in the old zone file has updated its resource record of type A, AAAA, NS, or MX in the new zone file, the domain is considered to be suspicious, and a respective client job is created for the domain. Each job has a priority, which defines the order of job to be processed by a client, i.e., the higher the sooner. The priority of a job is equal to the sum of predefined numbers of types of resource records that are changed, where the predefined numbers are 8, 4, 2, and 1 for A, NS, MX, and AAAA respectively. For example, if a domain has changed its A and NS records, the priority of the client job becomes 12.

When a client makes a job request to the server, the server responds with a few number of (five) jobs, which are either new or old with no response received for a certain period of time (1 hour) after it is sent to a

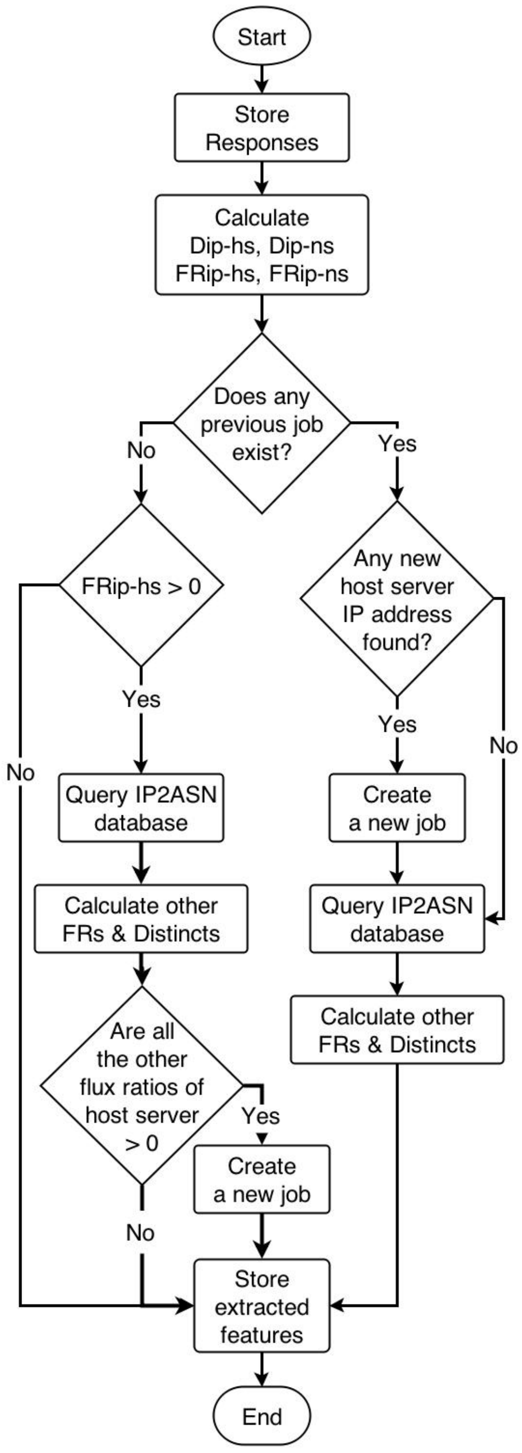
Start
Store Responses
Calculate Dip-hs, Dip-ns FRip-hs, FRip-ns
Does any previous job exist?
No
Yes
FRip-hs > 0
Any new host server IP address found?
Yes
No
Yes
No
Query IP2ASN database
Create a new job
Calculate other FRs & Distincts
Query IP2ASN database
Are all the other flux ratios of host server > 0
Calculate other FRs & Distincts
Yes
Create a new job
No
Store extracted features
End

Figure 8. *Flowchart – Processing responses*

client. As soon as a client has done processing a set of jobs, it returns the responses back to the server.

The server processes the responses for each client job as shown in figure 8. First, it stores the responses into the database. Then, for each domain, it counts distinct host server IP addresses (Dip-hs), distinct name server IP addresses (Dip-ns), and calculates IP address flux scores (FSip-hs and FSip-ns).

If the client job is the first job ever created for the domain, and host server IP address flux score (FSip-hs) is equal to zero, then the domain is considered to a normal domain, and the calculated features stored into the database. If the FSip-hs is greater than zero, the server maps all the IP addresses of the domain into their ASNs, organization names, network addresses, and country codes, which enables the server to calculate all the remaining features for the domain. If any of these newly calculated flux scores is greater than zero, the server creates a new client job for the domain. Otherwise, it just stores the calculated features into the database.

If the job is not the first job for the domain, and the IP addresses returned in the responses is not a subset of IP addresses collected by previous jobs, then the server creates new job for the domain. Otherwise, the domain is considered to be saturated which is a sign of a non-flux domain. In any cases, the server calculates all the other features, and stores them into the database.

The newly created job has a start date which indicates when the job should be sent to a client. The start date is set as current date time plus minimum TTL found in the returned responses.

5.2.2. ADAPT client

The main purpose of ADAPT client is to collect all the IP addresses related to the suspected domain by sending DNS queries to a number of DNS servers through Tor network. The ADAPT client gets the suspected domains from the ADAPT server in the form of client jobs, and returns the resolved IP addresses of the suspected domains in the form of client job responses. For a single domain, the ADAPT client sends a DNS query for a number of times depending on the intensity of the job. The following formula can be used to calculate total number of DNS queries sent for a single client job:

$$Nq = (Nrns + Nuns) * Nexit, \text{ where}$$

- Nq – the total number of DNS queries sent for a job,
- Nrns – the number of reliable name servers,
- Nuns – the number of unreliable name servers,
- Nexit – the number of exit nodes.

Reliable name servers are public RDNS servers, which are carefully chosen by the system administrator, and can be downloaded from the ADAPT server. And the number of reliable name servers to be probed is 3, 5, and 7 for intensity level 1, 2, and 3 respectively. On the other hand, unreliable name servers are downloaded from a 3rd party web-site such as [27], which provides a list of publicly accessible RDNS servers with their IP address and location information. Similar to probing reliable name servers, the client limits the number of unreliable name servers to be probed as 3, 5, and 7 for a job's intensity level 1, 2, and 3 respectively. The difference between probing reliable and unreliable name servers is, that, for unreliable name servers, the client tries to choose name servers from different countries, whereas for reliable name servers, the client randomly chooses name servers from the list of reliable name servers.

Furthermore, the intensity level of a job defines the minimum number of exit nodes from which, the client must probe the name servers again. For example, these numbers are 2, 3, and 5 for intensity level 1, 2, and 3 respectively. Sending the same DNS query to the same name server from different places or exit nodes could return a different DNS responses, because some public RDNS servers such as Google's public RDNS server take the client location into an account when resolving a domain.

To prevent the client from entering into an infinite loop, the client counts the NXDOMAIN responses and failed attempts to resolve the domain. If any of these numbers exceed their maximum limits, the client returns the responses back to the ADAPT server, even though the number of name servers that must be queried is not reached its limit. These maximum limits are also dependent upon the intensity level of the job, and they are equal to 3, 5, and 7 for intensity level 1, 2, and 3 respectively.

As the values of all of the above predefined numbers increase, the duration of processing a single job increases exponentially. Therefore, choosing proper values for those numbers are important for collecting enough data in a relatively short period of time.

5.3. Implementation Details

We have developed a prototype of ADAPT using Java and Groovy programming language in order to make it platform independent. Groovy is an agile and dynamic programming language for the Java Virtual Machine (JVM).

5.3.1. ADAPT server

ADAPT server consists of a database server and a web server. We use PostgreSQL as our database server, because it offers a variety of data types including IP address. Since most of our data consists of IP addresses, it helps the system to use the storage space efficiently. In order to decrease the required development time, we have implemented ADAPT server using Grails, which is an open source, full-stack, web application framework for the JVM.

A number of useful plug-ins are developed by 3rd parties for Grails to further decrease the development time. We have employed a several plug-ins such as “Spring Security” [28], “Quartz” [29], etc. The Spring Security plug-in provides an easy way to secure a web application with a mechanism for authenticating users and authorizing them to do their defined activities in the system. The Quartz plug-in helps us to implement a scheduled or a background task without any hassle. The use of Quartz scheduler comes in handy, when the ADAPT server downloads the zone files once a day, because it needs to be implemented as a scheduled task. Also, the Quartz jobs are used to execute long running tasks in the background, which helps the user not to wait for the task completion. Once, the zone files are downloaded, the server reads the zone files using the dnsjava [30] library to extract the suspicious domains.

Furthermore, Grails supports REST architecture [31], which simplifies the communication between server and client by using JSON [32] as a communication medium along with URL patterns that are representational of the underlying system, and HTTP methods. Therefore, the client job and client job responses are transferred as JSON string between client and server.

5.3.2. ADAPT client

ADAPT client is purely written in Java programming language. To make it portable, we employed HSQLDB [33], which offers a small, fast multi-threaded and transactional database engine with in-memory and disk-based tables and supports embedded mode. This database is used to store the information of reliable and unreliable name servers, since the number of these name servers easily exceeds a few hundreds.

In order to connect to Tor network, we used SilverTunnel-NG [34], which is a Java library that implements and encapsulates all the complex network protocol functions needed for anonymous communication over the Tor network. The current implementation of SilverTunnel-NG library lacks a full functionality of DNS resolution, i.e., it returns only the first IP address of the domain, when a domain name is requested to be resolved. Therefore, in order to obtain full DNS information of a domain, we extended a DNS resolver in the dnsjava library to use Tor network. This requires the IP address of a name server to be known prior to sending DNS query, which is why we are querying a publicly accessible DNS servers.

Since we are using Tor network and the suspected domains are required to be probed for several times, the system response time becomes relatively long. Therefore, to increase the throughput of ADAPT client, we implemented it as a multi-threaded console application. The main thread is responsible for creating all the sub threads, which are of two types: name server updater, and domain scanner. The main thread creates two name server updater sub threads for reliable name servers and unreliable name servers. These two threads update name server information independently from each other. The number of domain scanner sub threads is defined in the configuration file of the ADAPT client. Therefore, for each instance of an ADAPT client, n+3 number of threads are created, where n is the number of domain scanner threads.

5.4. Data Source

The ADAPT system uses the following data sources to detect malicious fast-flux domains:

- Domain zone files: Both Verisign, Inc. and ICANN provide zone data access services to law enforcement agents, IP attorneys, and researchers. TLD zone file access program from the Verisign [35] allows users to access to the TLD zone files for the .com, .net, and .name TLDs upon acceptance into the program. The zone files can be downloaded from Verisign's FTP server. Similarly, Centralized Zone Data Service (CZDS) from ICANN [36] offers access to the TLD zone files for over 300 TLDs. Most of the zone files can be download directly from the CZDS web site, however, a few TLDs require the user to go to their FTP server to download the zone files. The zone files get updated once in a day, hence the users are recommended to download the zone files only once in a day. This becomes the daily input to the ADAPT system.
- IP to ASN: Once the IP addresses of a domain is obtained, the ADAPT server maps the IP addresses into the ASN, network address, organization name, and country code, using 3rd party services such as [23]. To decrease the overhead on the 3rd party server, we used the DNS approach to query the IP to ASN database.
- Public RDNS servers: The ADAPT client downloads a list of public DNS servers from 3rd party web sites, and uses them as unreliable name servers. One of the web sites that offer a list of publicly accessible DNS servers as JSON string is [27]. A DNS server information consists of the IP address, host name, and location (country).

CHAPTER 6. ANALYSIS

We have conducted experiments on over 550,000 suspicious domains to analyze the output of our proposed detection technique ADAPT. The prototype of ADAPT system was set up on a single machine with specifications of 3.9GB RAM, AMD Turion(tm) II Dual-Core Mobile M520x2 CPU, and Ubuntu 14.04 64-bit operating system.

We started with extracting suspicious domains from two zone files of .net TLD, which are obtained on 9/25/2014 and 9/26/2014, and sizing around 1.4GB each. The extraction process took around 4 hours long. As a result, we have obtained 580,553 domains, that have changed any of their A, AAAA, MX, and NS resource records. Next, the ADAPT system started scanning the suspected domains with a single client with 35 threads and intensity level 1, which results a single domain to be probed from two different exit nodes against 6 DNS servers for each exit node. As a result of the first scan, the ADAPT distinguished the normal domains from the domains that employ RRDNS, CDN or FFN, which initiated the follow-up scans on the potential fast-flux domains to further distinguish domains that use FFNs. The results are summarized in the following sections.

6.1. First Scan Analysis

The ADAPT system has taken around 20 days to complete the first scan on all the 580,553 suspicious domains. As a result, we have got the features FSip-hs, FSip-ns, Dip-hs, and Dip-ns on each of the domains. The domains that have zero FSip-hs (host server IP address flux score) and zero FSip-ns (name server IP address flux score), are considered to be normal domains, because having zero IP address flux scores means that the domain is resolved into the same set of IP addresses from all the places. And, since the DNS is a worldwide distributed system, it is very hard to synchronize the DNS information on all the name servers in the DNS system in a short time (less than 24 hours). Figure 9 shows the ratio of the normal domains and potential fast-flux domains, where the potential fast-flux domains are total 221,931, resulting 38% of the all suspected domains.

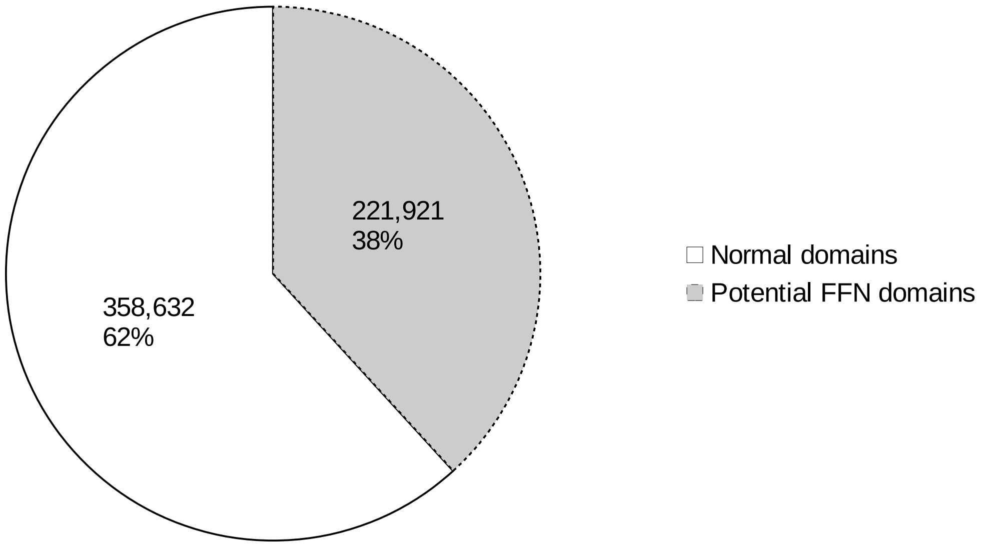

Figure 9. *Potential fast-flux domains*

If any of the features FSip-hs and FSip-ns of a domain is greater than zero, the ADAPT calculates the remaining features on the domain by mapping the IP addresses to the ASNs, network addresses, organization names, and country codes. Looking at the other flux scores, further, we can say a domain is normal, if all of them are equal to zero, which means the IP addresses resolved to the domain are all from the same network, same organization, and same country, which is a common characteristic of normal domains. Figure 10 depicts the ratio of normal domains and potential fast-flux domains after calculating FSasn-hs, FSasn-ns, FSnet-hs, FSnet-ns, FSorg-hs, FSorg-ns, FScc-ns, and FScc-hs.

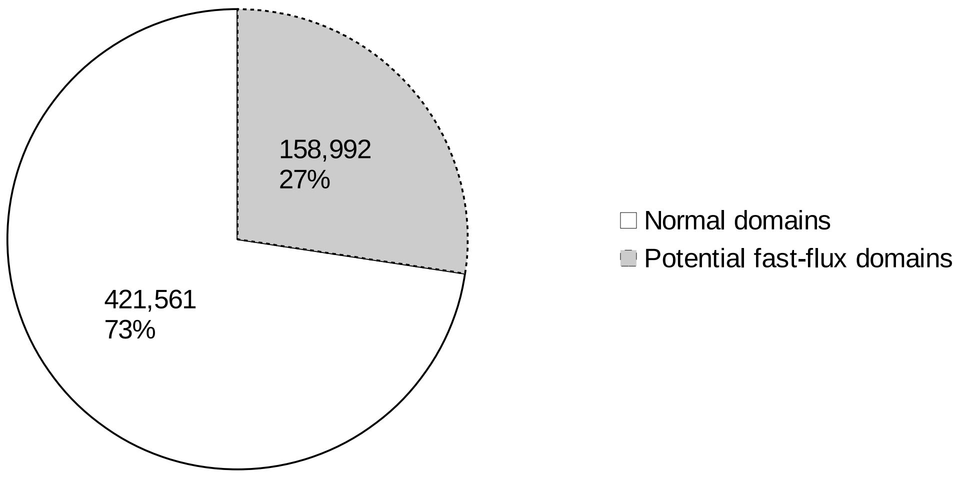

Figure 10. *Potential fast-flux domains after calculating other flux scores.*

On the remaining set of potential fast-flux domains, we counted the frequency of domains over the whole range of host server flux scores with 10% steps, which is shown in figure 11, where the number of domains are presented in a logarithmic scale. Looking at the graph, we can say most of the domains have low flux scores between 0 to 10, and the numbers decrease as the flux scores increase. This means, the host server flux scores could help us to detect malicious fast-flux domains.

Similarly, figure 12 shows the distributions of domains over the whole range of name server flux scores with 10% steps. However, in this case, all the domains are distributed relatively even across the whole range. Hence, we may say, the name server flux scores bring low information gain to the detection process. However, they can be used to further classify the fast-flux domains as single-flux or double-flux, because single-flux domains change their host server IP addresses, and the double-flux domains change their host server IP addresses as well as name server IP addresses. Therefore, not considering any of name server flux scores in the detection of fast-flux domain could help us to decrease the number of domains need to be scanned in the follow-up scans.

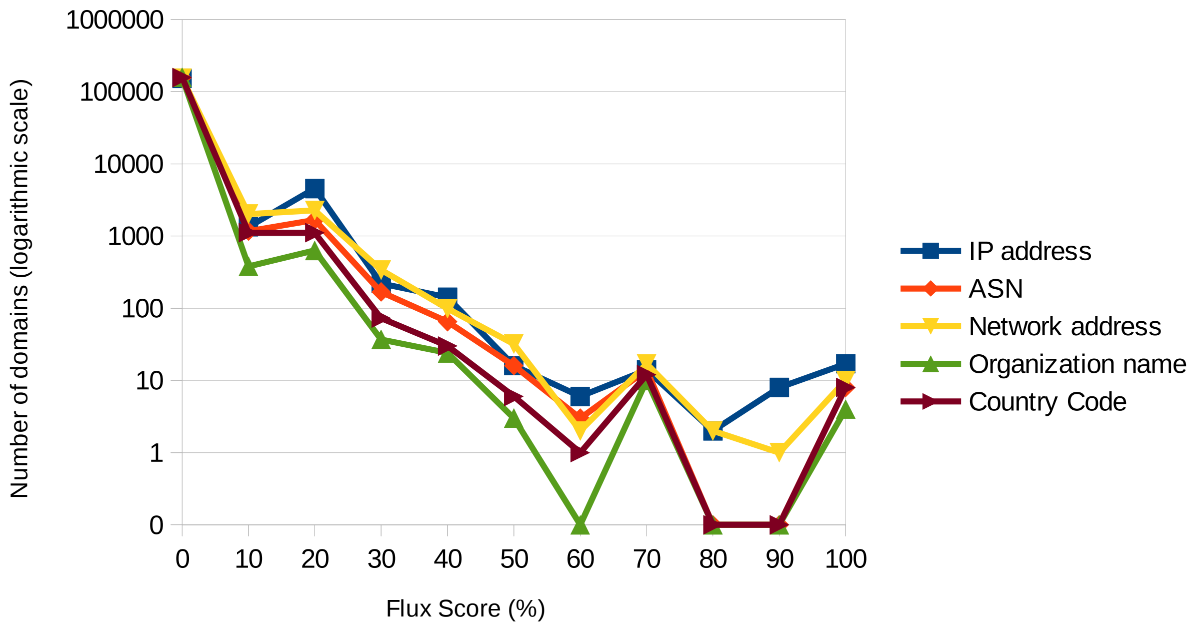

Figure 11. *Flux Score Distribution – Host server.*

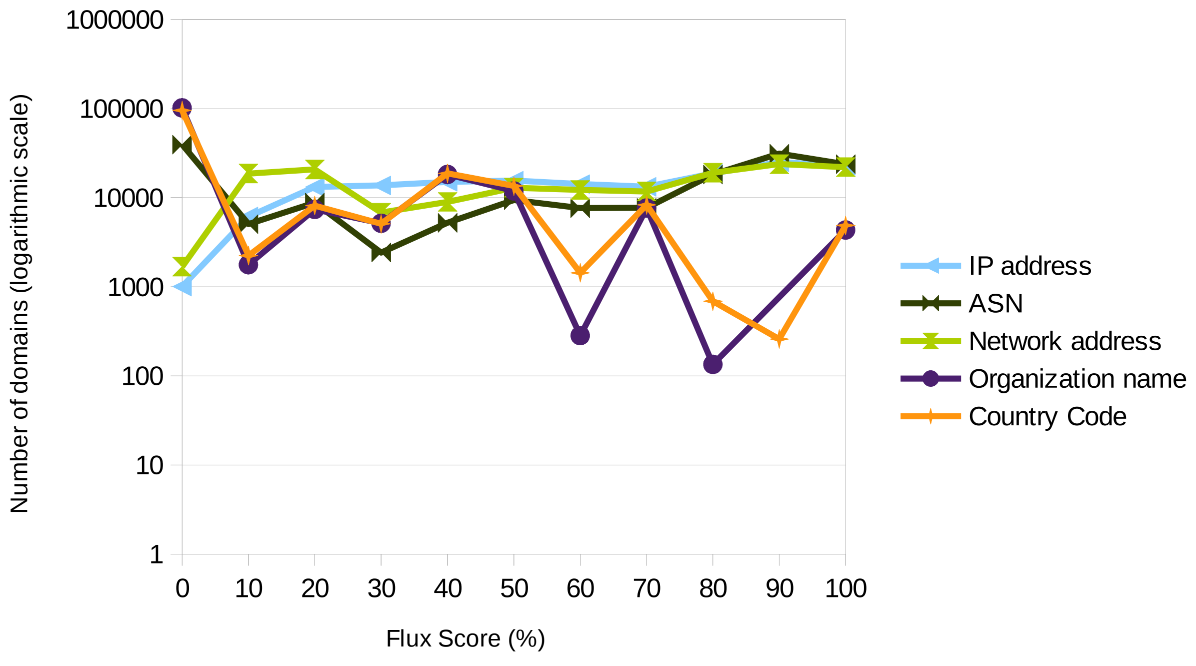

Figure 12. *Flux Score Distribution – Name server.*

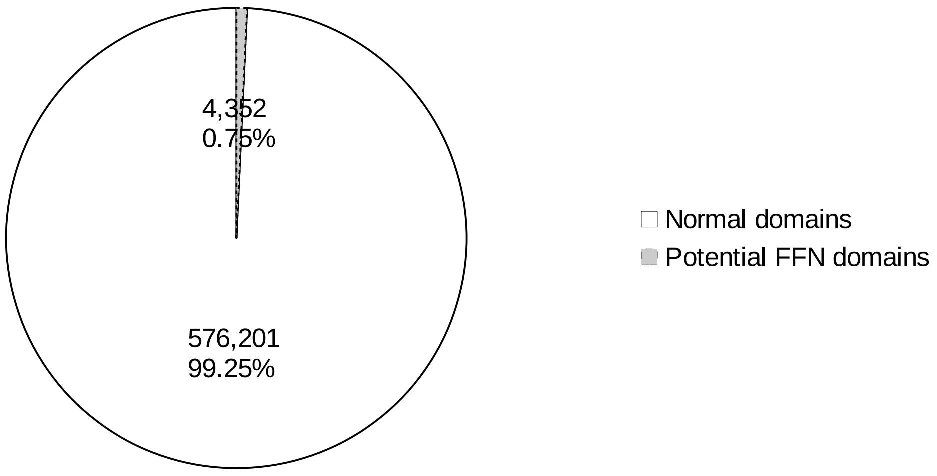

Figure 13. *Potential fast-flux domains after considering only host server flux scores.*

Figure 13 shows the number of potential fast-flux domains in the all suspected domains by filtering with the following conditions: FSip-hs > 0, and any of FSorg-hs, FSnet-hs, FScc-hs, and FSasn-hs is greater than zero. This results the potential fast-flux domains to be only 0.75% of the all suspected domains, which is 4,352. To further investigate, we need to look at the data obtained by follow-up scans.

6.1.1. NXDOMAIN analysis

When a domain name is unable to be resolved using the DNS, a response of type NXDOMAIN is returned, which indicates the domain does not exist in the Internet. Since malicious domains tend to have shorter lifespan due to takedown by authorities, they often resolve to NXDOMAIN responses after some time. Therefore, it is important to look at the NXDOMAIN responses collected by the ADAPT to measure the possible takedown of malicious domains that the ADAPT has missed the chance to detect. The domains that resolved to NXDOMAIN responses can be categorized as the domains that resolved to only NXDOMAIN, and the domains that resolved to NXDOMAIN at least once. And their ratios in the all suspicious domains are shown in figure 14 and 15, respectively.

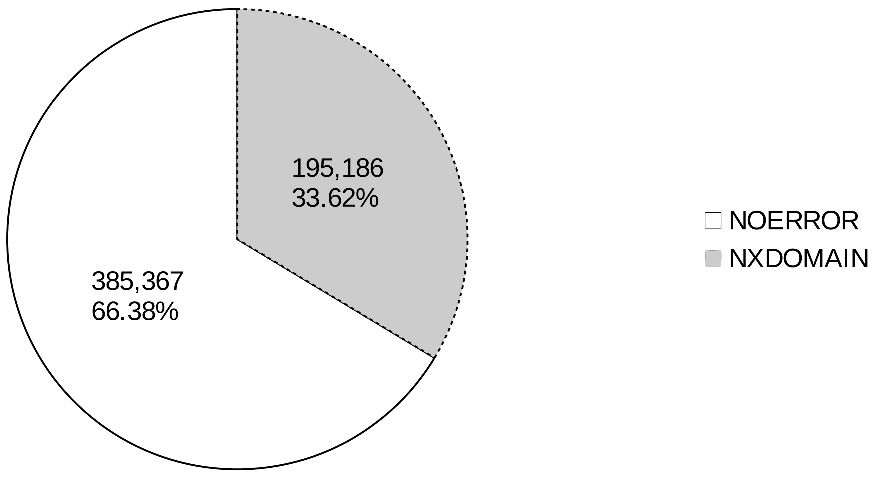

Figure 14. *Domains that resolved to only NXDOMAIN responses.*

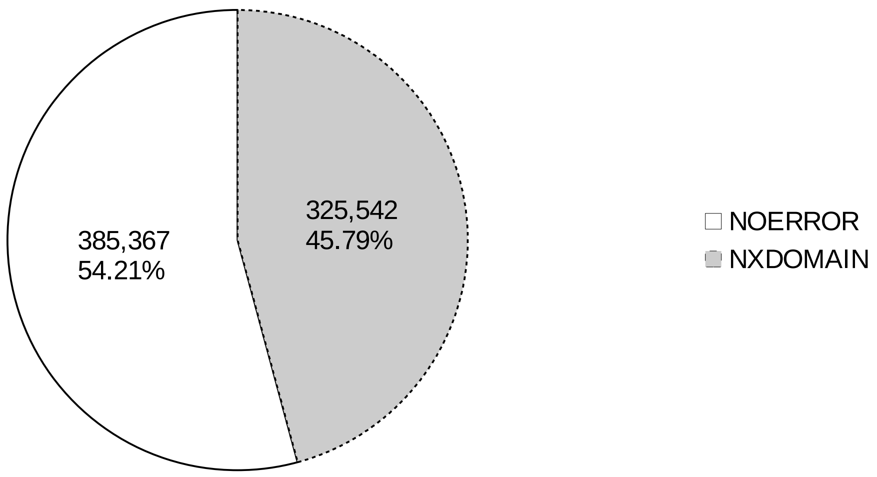

Figure 15. *Domains that resolved to at least one NXDOMAIN response.*

Also, figure 16 shows the distribution of domains that resolved to NXDOMAIN responses across the duration of first scan. From the graph, we can see that the number of domains that resolve to NXDOMAIN response increases generally as the day goes by. Therefore, we suspect that many malicious domains have been taken down already by the time of completion of our first scan.

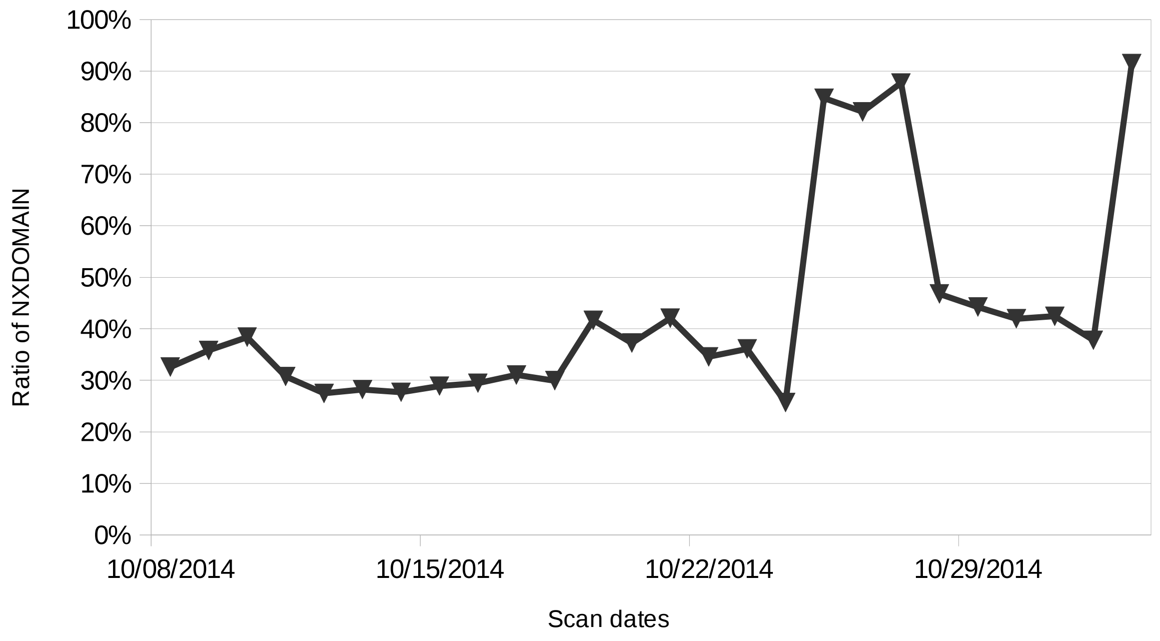

Figure 16. *Distribution of domains that resolved to at least one NXDOMAIN response.*

6.2. Follow-up Scan Analysis

After the second scan, total 364 domains out of 4352 domains were resolved to new IP addresses, which is shown in the figure 17. Looking at the results manually, we observed that the domains hosted on Amazon Web Service (AWS) have very high IP flux scores and low ASN flux scores. And the number of their distinct host server IP addresses was over 250 after only the second scan. Furthermore, the domains which have second highest IP flux scores were the domains which offer Virtual Private Network (VPN) services to individuals to provide anonymity. They claim that they are fully compliant with the law and the servers used in the service belong to private and independent individuals from different parts of the globe, which might be the reason of having higher flux scores. Moreover, we encountered a number of domains which host the same looking content or no content at all. We suspect that they could be fast-flux domains in their infancy.

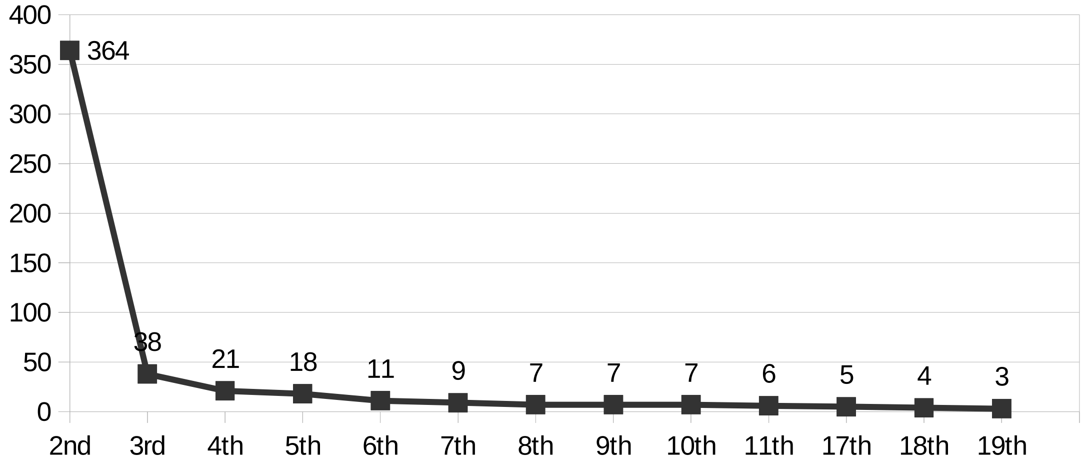

Figure 17. *Potential fast-flux domains remained after each iteration of the follow-up scan.*

After the 19th iteration of the follow-up scan, only the domains hosted on AWS kept being resolved to new IP addresses. Each iteration of the scan is triggered after TTL seconds since the previous scan is completed, which increases the probability of discovering a new IP address of the domain. However, from the observation, we found the assumption that malicious fast-flux domains update their IP address after every TTL value could be false, and it could enable the attacker to evade the ADAPT if the malicious fast-flux domain updates its IP address only when its one of the flux-agents goes off-line, which is usually longer than the TTL value.

Furthermore, we counted the IP addresses shared between host servers and name servers for each of the domains which have been scanned in the follow-up scans. Out of 364 domains, 252 domains are found to be sharing IP addresses between its host servers and name servers. Out of those 252 domains, 201 domains have a name starting with either “ns” or “dns”, which could indicate they themselves are name servers. The remaining 51 domains consist of VPN service providers, which had very high IP share ratio, and some other domains which host empty content or default web server banner. Interestingly, the domains that are found to be hosted on AWS, have not shared any IP address between its host servers and name servers. This observation is supported by one of new trends reported in [21]. Therefore, we believe that the IP address share ratio could be used as a detection feature in the future work.

6.3. Summary of Major Findings

By analyzing the scan result, we made the following conclusions:

1. The flux scores of name server seem to have low influence on fast-flux domain detection. However, they may be used to further classify a fast-flux domain as single-flux or double-flux.
2. Since malicious domains are known to have short life time, the detection process should be fast enough.
3. Domains hosted on benign CDNs such as AWS update their host server IP addresses more than any other domains.
4. VPN service providers use the same technique as the fast-flux. However, their nodes consist of machines provided by individuals who has willingness to serve the VPN users.
5. Malicious fast-flux domains can update their IP addresses only when their flux-agents go off-line.
6. IP address share ratio between host server and name server could be used as a detection feature.
7. The ADAPT is able to detect fast-flux domains in their infancy, even if it has not been involved with any malicious activity.

CHAPTER 7. DISCUSSION

We suspect many malicious fast-flux domains had been taken down by other detection systems in the Internet before the ADAPT has detected them, because the initial suspicious domains were obtained between 9/25/2014 and 9/26/2014, and the whole scanning process completed on 11/6/2014. This presumption is supported by the number of domains that resolved into NXDOMAIN responses in Section 6.1. However, the ADAPT has found a few potential fast-flux domains, that shows it has a high potential doing better job than the existing systems, if it could complete all the scanning process before the next suspicious domains fed into the system on the next day. Apparently, for that, the ADAPT has to have sufficient number of clients that can handle all the load in one day. In other words, as the number of suspicious domains grows, the more number of clients must be employed.

One of the new trends reported in [21], i.e., the CDNs are using lower TTL values, whereas FFNs are using higher TTL values, is confirmed by our data analysis. And, we observed that the domains hosted on CDNs updated their host server IP addresses more often than any other domains, however the numbers of their distinct ASNs were very low, close to 1. Although, a FFN could mimic this characteristic by using flux-agents from a single ASN to evade the detection system, it will be a huge trade-off for the FFN, because the number of flux-agents will be decreased significantly as the number of ASNs get decreased. Therefore, we could use this characteristic to distinguish CDNs from FFNs.

We observed that some VPN service providers use the same technique as fast-flux. In order to differentiate them from malicious fast-flux domains, the content hosted on the domains can be compared. We can calculate the similarity of the contents of the domains, i.e., the Feature

20 in Table 1 in Chapter 4, using the string kernel method as proposed in [4]. Therefore, it is necessary to employ this feature in the future work. .

Furthermore, a malicious fast-flux domain can evade the ADAPT detection system by updating its IP addresses only when one of its flux-agents goes off-line rather than after every TTL seconds. The feature 16 in Table 1 in Chapter 4, guilt by association score, can be employed to detect a malicious fast-flux domain that uses a set of IP addresses that had been used previously by other malicious fast-flux domains. According to [21], the malicious fast-flux domains share their IP addresses with each other, thus, we believe that the guilt by association score can increase the detection rate. However, before start using this feature, the detection system needs initial historical data set, which can be obtained by scanning blacklisted malicious domains.

Apparently, the guilt by association score cannot detect a malicious fast-flux domain that does not share any IP address with other malicious domains. Therefore, we need to add some other detection features, such as IP address share ratio between host servers and name servers. To evade this detection feature, the FFNs are required to decrease the number of host servers further.

Finally, using a predefined list of public RDNS servers was not our initial idea. While implementing the prototype system, we have found out the current version of Java library, SilverTunnel-NG does not support full DNS resolution on a domain, rather it just returns a single IP address of a domain that is resolved on the local RDNS server of the current exit node in the Tor network. Consequently, in order to get the full DNS information of a domain using the current version of SilverTunnel-NG, we had to query public RDNS servers, which could affect the system performance. Therefore, we recommend to find a solution to this problem, and start taking the full benefits of Tor network in the future work.

CHAPTER 8. SUMMARY AND FUTURE WORK

8.1. Summary

Considering an emerging security threat of malicious fast-flux domains, and inspired by the work of researchers on DNS analysis for malicious domain detection, we have proposed an Anonymous, Distributed, Active Probing Technique (ADAPT) to detect malicious fast-flux domains analyzing DNS traffic, and developed a prototype of ADAPT detection system. Our contributions to the field of malicious fast-flux domain detection using DNS analysis are as follows:

- A prototype of an active probing technique, which is anonymous, has worldwide view on a domain, and opens the door to detect in-the-wild malicious fast-flux domains.
- New detection features such as flux scores, and IP address share ratio between host servers and name servers.
- New findings on FFN characteristics and domains which have the same characteristics with FFNs.

In Chapter 2, we briefly introduced the DNS system, its deployment variations, and fast-fluxing for technical background. And, in Chapter 3, we included a literature survey of existing researches on fast-flux domain detection to show their advantages and drawbacks. Consequently, we summarized their common drawbacks as follows: being reactive, not anonymous, partial view on the suspected domain, and unable to detect domains that use well-developed FFN that mimics benign CDN characteristics.

In Chapter 4, we defined the problem statement along with the spatial and temporal DNS traffic data structure that we can obtain by actively probing DNS servers. Subsequently, we extracted 21 features from the existing researches that are covered in Chapter 3.

In Chapter 5, we detailed our proposed approach starting with its high level system architecture. The ADAPT system has 3 tier client-server architecture, which makes it scalable. Even though, the ADAPT can scan a domain on demand, the main input to the ADAPT is zone file, and we briefly covered its benefit for early detection. We introduced 10 new features derived from a new formula we defined for the flux score. Also, we explained the two reasons to employ Tor network in the data collection process, which are anonymity and world-wide distribution. Then, the functional details section covered the general workflow of data collection and domain classification processes, where the ADAPT client collects DNS information for the suspected domain from a number of different DNS servers through Tor network, and the ADAPT server analyzes the collected data, and makes decision on whether the domain should be scanned further to prove it is a fast-flux domain. In the implementation details section, we specified each of the technologies and libraries we used in the prototype system as follows: PostgreSQL for efficient data storage since most of our data consist of IP address and PostgreSQL has a data type for storing IP address, Grails for fast development, Java for platform independency, SilverTunnel-NG library for connecting to Tor network, and dnsjava library for handling DNS packets. In the end of the chapter, we referenced the data sources the ADAPT system uses, such as zone files from Verisign, Inc. and ICANN, a list of public RDNS servers from [27], and IP to ASN mapping from [23].

In Chapter 6, we analyzed the data obtained by the prototype system. From the first scan, we distinguished 4,352 potential fast-flux domains out of 580,553 suspicious domains obtained from .net TLD zone files. First, the IP address flux scores FSip-hs, FSip-ns are calculated for each domain, and as a result 358,632 domains out of 580,553 found to have zero FSip-hs, FSip-ns. Then, for each of the remaining 221,921 domains, the other flux scores FSasn-hs, FSasn-ns, FSnet-hs, FSnet-ns, FSorg-hs, FSorg-ns, FScc-ns, and FScc-hs were calculated, and they were zero for 62,929 domains. Further, we analyzed the flux score distribution of host server and name server on all the remaining 158,992 domains, which showed that the domains are distributed relatively even over the whole range of name server flux scores. Therefore, we excluded the name server flux scores from the filter and only used host server flux scores, which resulted 4,352 domains to be found as potential fast-flux domains. Finally, looking at the results of the follow-up scans, we observed a few new findings on the fast-flux domains, and made seven conclusions in the end of the chapter.

In Chapter 7, we discussed the limitations of our current work, and solutions to those limitations. The limitations can be seen as resource based, new findings based, and technology based. The resource based limitation involves the speed of data collection, the new findings based limitations involve the features used in the detection process, and technology based limitation involves the inabilities of current versions of the technologies and libraries that are used in the prototype implementation.

8.2. Future Work

In the prototype of ADAPT, we used only 20 detection features, and we believe the detection accuracy and speed can be increased by utilizing extra features such as the features 16, 20 in Table 1 in Chapter 4, and IP address sharing ratio between name server and host server along with a proper machine learning algorithm similar to [5], [10], [15].

As mentioned in Chapter 7, we need to find a way to probe the local RDNS servers from the exit nodes in the Tor network. This will increase the system performance significantly. Also, during the scanning process, we found that the current version of Grails has a memory leak problem [37] which results the out of memory exception when a long running task is executed on the server. This should be fixed, or we need a better Java web application framework to develop the detection system.

REFERENCES

[1] "Canadian Pharmacy - Spamwiki." [Online]. Available: http://spamtrackers.eu/wiki/index.php/Canadian_Pharmacy. [Accessed: 14-Oct-2014].

[2] "FortiGuard.com | Canadian Pharmacy." [Online]. Available: http://www.fortiguard.com/legacy/analysis/canadianpharmacy.html. [Accessed: 14-Oct-2014].

[3] "New Zeus Gameover Employs DGA and Fast Flux Techniques | Threat Encyclopedia - Trend Micro Inc." [Online]. Available: http://www.trendmicro.com/vinfo/us/threat-encyclopedia/spam/578/new-zeus-gameover-employs-dga-and-fast-flux-techniques. [Accessed: 14-Oct-2014].

[4] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and Detecting Fast-Flux Service Networks.," *NDSS*, 2008.

[5] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces," *2009 Annu. Comput. Secur. Appl. Conf.*, pp. 311–320, Dec. 2009.

[6] T. K. -, H. C. -, and C. C. -, "Detecting and Analyzing Fast-Flux Service Networks," *Int. J. Adv. Inf. Sci. Serv. Sci.*, vol. 4, no. 10, pp. 183–190, Jun. 2012.

[7] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, "Genetic-based real-time fast-flux service networks detection," *Comput. Networks*, vol. 57, no. 2, pp. 501–513, Feb. 2013.

[8] Z. Berkay Celik and S. Oktug, "Detection of Fast-Flux Networks using various DNS feature sets," *2013 IEEE Symp. Comput. Commun.*, pp. 000868–000873, Jul. 2013.

[9] Z. Futai, Z. Siyu, and R. Weixiong, "Hybrid detection and tracking of fast-flux botnet on domain name system traffic," *China Commun.*, vol. 10, no. 11, pp. 81–94, Nov. 2013.

[10] S. Martinez-Bea, S. Castillo-Perez, and J. Garcia-Alfaro, "Real-time malicious fast-flux detection using DNS and bot related features.," *PST*, 2013.

[11] C. Chen, M. Huang, and Y. Ou, "Detecting Hybrid Botnets with Web Command and," vol. 5, no. 2, pp. 263–274, 2014.

[12] P. V. Mockapetris, "Domain names: Concepts and facilities," 1983. [Online]. Available: https://tools.ietf.org/html/rfc882. [Accessed: 30-Mar-2014].

[13] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49. pp. 101–106, 2006.

[14] A. Shaikh, R. Tewari, and M. Agrawal, “On the effectiveness of DNS-based server selection,” *Proc. IEEE INFOCOM 2001. Conf. Comput. Commun. Twent. Annu. Jt. Conf. IEEE Comput. Commun. Soc. (Cat. No.01CH37213)*, vol. 3, 2001.

[15] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, “FluXOR: Detecting and monitoring fast-flux service networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5137 LNCS, pp. 186–206.

[16] S. Huang and H. Lee, “Fast-flux Service Network Detection Based on Spatial Snapshot Mechanism for Delay-free Detection Categories and Subject Descriptors,” pp. 101–111, 2010.

[17] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, “Real-Time Detection of Fast Flux Service Networks,” *2009 Cybersecurity Appl. Technol. Conf. Homel. Secur.*, pp. 285–292, Mar. 2009.

[18] C. Hsu, C. Huang, and K. Chen, “Fast-flux bot detection in real time,” *Recent Adv. Intrusion Detect.*, 2010.

[19] B. N. Al-Duwairi and A. T. Al-Hammouri, “Fast Flux Watch: A mechanism for online detection of fast flux networks,” *J. Adv. Res.*, vol. 5, no. 4, pp. 473–479, Jul. 2014.

[20] K. D. McGrath, A. Kalafut, and M. Gupta, “Phishing infrastructure fluxes all the way,” *IEEE Secur. Priv.*, vol. 7, pp. 21–28, 2009.

[21] W. Xu, X. Wang, and H. Xie, “New Trends in FastFlux Networks,” *media.blackhat.com*, 2013.

[22] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, “EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.,” *NDSS*, pp. 1–17, 2011.

[23] “IP to ASN Mapping - Team Cymru.” [Online]. Available: http://www.team-cymru.org/Services/ip-to-asn.html. [Accessed: 30-Oct-2014].

[24] S. Hao, G. Tech, and N. Feamster, “Monitoring the Initial DNS Behavior of Malicious Domains Categories and Subject Descriptors,” pp. 269–278, 2011.

[25] J. Spring, L. Metcalf, and E. Stoner, “Correlating domain registrations and DNS first activity in general and for malware,” *Secur. Trust. Internet …*, no. October, pp. 1–4, 2011.

[26] M. Bhuiyan and A. Mohaisen, “Characterization of the Dynamics and Interactions of Domain Names and Name Server,” … *Netw. Secur. (…*, pp. 265–266, 2013.

[27] "Public DNS Server List." [Online]. Available: http://public-dns.tk/. [Accessed: 03-Nov-2014].

[28] "Grails Plugin: Spring Security Core Plugin." [Online]. Available: http://grails.org/plugin/spring-security-core. [Accessed: 03-Nov-2014].

[29] "Grails Plugin: Quartz plugin for Grails." [Online]. Available: http://grails.org/plugin/quartz. [Accessed: 03-Nov-2014].

[30] "dnsjava." [Online]. Available: http://www.xbill.org/dnsjava/. [Accessed: 03-Nov-2014].

[31] "Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)." [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Accessed: 03-Nov-2014].

[32] "JSON." [Online]. Available: http://www.json.org/. [Accessed: 03-Nov-2014].

[33] "HSQLDB." [Online]. Available: http://hsqldb.org/. [Accessed: 03-Nov-2014].

[34] "SilverTunnel-NG | SourceForge.net." [Online]. Available: http://sourceforge.net/projects/silvertunnel-ng/. [Accessed: 03-Nov-2014].

[35] "Zone File Information and TLD Zone Files - Verisign." [Online]. Available: http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/zone-file-information/index.xhtml. [Accessed: 03-Nov-2014].

[36] "CZDS - Resources - ICANN." [Online]. Available: https://www.icann.org/resources/pages/czds-2014-03-03-en. [Accessed: 03-Nov-2014].

[37] "[GRAILS-5823] JAR Locking / Resource Clean Up - Grails JIRA." [Online]. Available: https://jira.grails.org/browse/GRAILS-5823. [Accessed: 06-Nov-2014].

APPENDIX

The source code of the prototype of ADAPT can be downloaded from:

http://home.engineering.iastate.edu/~guan/adapt/