EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis

Leyla Bilge*, Engin Kirda*§, Christopher Kruegel‡, and Marco Balduzzi*

* Institute Eurecom, Sophia Antipolis
{bilge,kirda,balduzzi}@eurecom.fr

§ Northeastern University, Boston
ek@ccs.neu.edu

§ University of California, Santa Barbara
chris@cs.ucsb.edu

Abstract

The domain name service (DNS) plays an important role in the operation of the Internet, providing a two-way mapping between domain names and their numerical identifiers. Given its fundamental role, it is not surprising that a wide variety of malicious activities involve the domain name service in one way or another. For example, bots resolve DNS names to locate their command and control servers, and spam mails contain URLs that link to domains that resolve to scam servers. Thus, it seems beneficial to monitor the use of the DNS system for signs that indicate that a certain name is used as part of a malicious operation.

In this paper, we introduce EXPOSURE*, a system that employs large-scale, passive DNS analysis techniques to detect domains that are involved in malicious activity. We use* 15 *features that we extract from the DNS traffic that allow us to characterize different properties of DNS names and the ways that they are queried.*

Our experiments with a large, real-world data set consisting of 100 billion DNS requests, and a real-life deployment for two weeks in an ISP show that our approach is scalable and that we are able to automatically identify unknown malicious domains that are misused in a variety of malicious activity (such as for botnet command and control, spamming, and phishing).

1 Introduction

The Domain Name System (DNS) is a hierarchical naming system for computers, services, or any resource connected to the Internet. Clearly, as it helps Internet users locate resources such as web servers, mailing hosts, and other online services, DNS is one of the core and most important components of the Internet. Unfortunately, besides being used for obvious benign purposes, domain names are also popular for malicious use. For example, domain names are increasingly playing a role for the management of botnet command and control servers, download sites where malicious code is hosted, and phishing pages that aim to steal sensitive information from unsuspecting victims.

In a typical Internet attack scenario, whenever an attacker manages to compromise and infect the computer of an end-user, this machine is silently transformed into a bot that listens and reacts to remote commands that are issued by the so-called botmaster. Such collections of compromised, remotely-controlled hosts are common on the Internet, and are often used to launch DoS attacks, steal sensitive user information, and send large numbers of spam messages with the aim of making a financial profit.

In another typical Internet attack scenario, attackers set up a phishing website and lure unsuspecting users into entering sensitive information such as online banking credentials and credit card numbers. The phishing website often has the look and feel of the targeted legitimate website (e.g., an online banking service) and a domain name that sounds similar.

One of the technical problems that attackers face when designing their malicious infrastructures is the question of how to implement a reliable and flexible server infrastructure, and command and control mechanism. Ironically, the attackers are faced with the same engineering challenges that global enterprises face that need to maintain a large, distributed and reliable service infrastructure for their customers. For example, in the case of botnets, that are ar-

guably one of the most serious threats on the Internet today, the attackers need to efficiently manage remote hosts that may easily consists of thousands of compromised end-user machines. Obviously, if the IP address of the command and control server is hard-coded into the bot binary, there exists a single point of failure for the botnet. That is, from the point of view of the attacker, whenever this address is identified and is taken down, the botnet would be lost.

Analogously, in other common Internet attacks that target a large number of users, sophisticated hosting infrastructures are typically required that allow the attackers to conduct activities such as collecting the stolen information, distributing their malware, launching social engineering attempts, and hosting other malicious services such as phishing pages.

In order to better deal with the complexity of a large, distributed infrastructure, attackers have been increasingly making use of domain names. By using DNS, they acquire the flexibility to change the IP address of the malicious servers that they manage. Furthermore, they can hide their critical servers behind proxy services (e.g., using Fast-Flux [36]) so that their malicious server is more difficult to identify and take down.

Using domain names gives attackers the flexibility of migrating their malicious servers with ease. That is, the malicious "services" that the attackers offer become more "fault-tolerant" with respect to the IP addresses where they are hosted.

Our key insight in this paper is that as malicious services are often as dependent on DNS services as benign services, being able to identify malicious domains as soon as they appear would significantly help mitigate many Internet threats that stem from botnets, phishing sites, malware hosting services, and the like. Also, our premise is that when looking at large volumes of data, DNS requests for benign and malicious domains should exhibit enough differences in behavior that they can automatically be distinguished.

In this paper, we introduce a passive DNS analysis approach and a detection system, EXPOSURE, to effectively and efficiently detect domain names that are involved in malicious activity. We use 15 features (9 of which are novel and have not been proposed before) that allow us to characterize different properties of DNS names and the ways that they are used (i.e., queried).

Note that researchers have used DNS before as a way to analyze, measure and estimate the size of existing botnets in the past (e.g., [21, 22, 34]). Some solutions have then attempted to use DNS traffic to detect malicious domains of a certain type (e.g., [30, 36]). However, all these approaches have only focused on specific classes of malware (e.g., only malicious Fast-Flux services). Our approach, in comparison, is much more generic and is not only limited to certain classes of attacks (e.g., only botnets).

In our approach, based on features that we have identified and a training set that contains known benign and malicious domains, we train a classifier for DNS names. Being able to passively monitor real-time DNS traffic allows us to identify malware domains that have not yet been revealed by pre-compiled blacklists. Furthermore, in contrast to active DNS monitoring techniques (e.g., [36]) that probe for domains that are suspected to be malicious, our analysis is stealthy, and we do not need to trigger specific malicious activity in order to acquire information about the domain. The stealthy analysis that we are able to perform has the advantage that our adversaries, the cyber-criminals, have no means to block or hinder the analysis that we perform (in contrast to approaches such as in [36]).

To date, only one system has been proposed that aims to detect malicious domains generically using passive DNS analysis. In a concurrent and independent work that was very recently presented by Antonakakis et al. [11], the authors present Notos. Notos dynamically assigns reputation scores to domain names whose maliciousness has not been discovered yet. In comparison, our approach is not dependent on large amounts of historical maliciousness data (e.g., IP addresses of previously infected servers), requires less training time, and unlike Notos, is also able to detect malicious domains that are mapped to a new address space each time and never used for other malicious purposes again.

In our offline experiments, we have applied EXPOSURE to a large, real-world data set collected over a period of two and a half months. The data that we used for the initial training consists of DNS traffic from the Security Information Exchange (SIE) [7] that shares with us the real-time response data from authoritative name servers located in North America and in Europe. These sensors receive large amounts of data. In fact, during the analysis period of 2.5 months, our system monitored and analyzed more than 100 billion DNS queries that targeted 4.8 million distinct domain names.

Furthermore, in order to determine the feasibility of our detection approach in real-life, we used EXPOSURE to train on and monitor the DNS traffic of a commercial ISP that supports more than 30,000 clients. We were able to identify more than 3000 new malicious domains that were previously unknown to our system and were not in our training set. Moreover, we have cross-checked our detection results with public services such as malwareurl.com, McAffee Site Advisor, and Norton Safe Web. The experimental results we present in this paper show that our approach is scalable, and that we are able to automatically identify domain names that are misused in a variety of malicious activity.

In summary, our paper makes the following contributions:

- We present a novel analysis technique for the detection of malicious domains that is based on passive

DNS request analysis. Our technique does not rely on prior knowledge about the kind of service the malicious domain provides (e.g., phishing, Fast-Flux services, spamming, botnets that use a domain generation algorithm, etc.). This is significantly different from existing techniques that only target Fast-Flux domains used in botnet operations. Furthermore, our approach requires less training time, and less training data than Notos [11], and does not have some of its limitations.

- We present 15 behavioral features that our system uses in the the identification of malicious domains. Of these features, 9 have not been proposed before in previous research.

- We describe the implementation of our real-time detection prototype system which we call EXPOSURE. We used large volumes of DNS data that we collected over a two and a half month period as the offline data set for EXPOSURE. During this period, we recorded 100 billion DNS queries that resulted in 4.8 million distinct domain names. Furthermore, we deployed EXPOSURE in real-life in an ISP and used it to monitor the DNS traffic of 30,000 clients. Our experimental results show that the technique we propose is scalable, and is able to accurately distinguish between malicious and benign domains with a low false positive rate.

2 Overview of the Approach

The goal of EXPOSURE is to detect malicious domains that are used as part of malicious operations on the Internet. To this end, we perform a passive analysis of the DNS traffic that we have at our disposal. Since the traffic we monitor is generated by real users, we assume that some of these users are infected with malicious content, and that some malware components will be running on their systems. These components are likely to contact the domains that are found to be malicious by various sources such as public malware domain lists and spam blacklists. Hence, by studying the DNS behavior of known malicious and benign domains, our goal was to identify distinguishable generic features that are able to define the maliciousness of a given domain.

2.1 Extracting DNS Features for Detection

Clearly, to be able to identify DNS features that allow us to distinguish between benign and malicious domains, and that allow a classifier to work well in practice, large amounts of training data are required. As the offline dataset, we recorded the recursive DNS (i.e., RDNS) traffic from Security Information Exchange (SIE) [7]). We performed offline analysis on this data and used it to determine DNS features that can be used to distinguish malicious DNS features from benign ones. The part of the RDNS traffic we used as initial input to our system consisted of the DNS answers returned from the authoritative DNS servers to the RDNS servers. An RDNS answer consists of the name of the domain queried, the time the query is issued, the duration the answer is required to be cached (i.e., TTL) and the list of IP addresses that are associated with the queried domain. Note that the RDNS servers do not share the information of the DNS query source (i.e. the IP address of the user that issues the query) due to privacy concerns.

By studying large amounts of DNS data, we defined 15 different features that we use in the detection of malicious domains. 6 of these features have been used in previous research(e.g., [29, 30, 36]), in particular in detecting malicious Fast-Flux services or in classifying malicious URLs [27]. The features that we use in the detection and our rationale for selecting these features are explained in detail in Section 3.

2.2 Architecture of EXPOSURE

Figure 1 gives an overview of the system architecture of the EXPOSURE. The system consists of five main components:

The first component, the Data Collector, records the DNS traffic produced by the network that is being monitored.

The second component is the Feature Attribution component. This component is responsible for attributing the domains that are recorded to the database with the features that we are looking for in the DNS traffic.

The third component, the Malicious and Benign Domains Collector, works independent of, and in parallel to the Data Collector Module. It collects domains that are known to be benign or malicious from various sources. Our benign domains sets are composed of information acquired from Alexa [4] and a number of servers that provide detailed WHOIS [2] data. In contrast, the malicious domain set is constructed from domains that have been reported to have been involved in malicious activities such as phishing, spamming, and botnet infections by external sources such as malwaredomains.com, Phishtank ([31]), and malware analyzers such as Anubis [13]). Note that these lists are constantly updated, and become even more comprehensive over time. The output of the Malicious and Benign Domains Collector is used to label the output of the Feature Attribution component.

Once the data is labeled, the labeled set is fed into the fourth component: The Learning Module. This module trains the labeled set to build malicious domain detection models. Consequently, these models, and the unlabeled domains, become an input to the fifth component: The Classi-

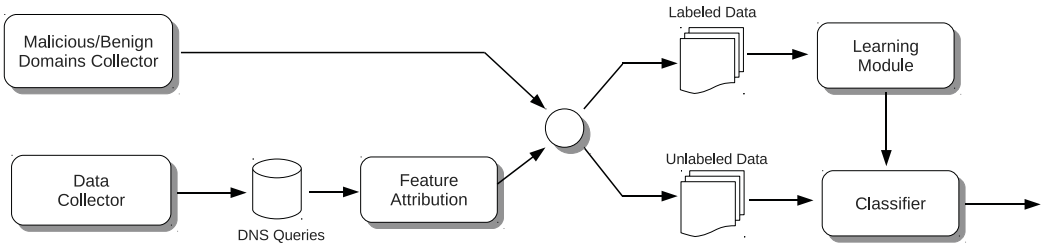


Figure 1: Overview of EXPOSURE

fier.

The Classifier component takes decisions according to the detection models produced by the Learning component so that the unlabeled domains are grouped into two classes: domains that are malicious, and those that are benign.

2.3 Real-Time Deployment

The deployment phase of EXPOSURE consists of two steps. In the first step, the features that we are interested in are monitored and the classifier is trained based on a set of domains that are known to be benign or malicious. In a second step, after the classifier has been trained, the detection starts and domains that are determined to be suspicious are reported. Note that after an initial period of seven days of training[1], the classifier is retrained every day. Hence, EXPOSURE can constantly keep up with the behavior of new malware.

3 Feature Selection

To determine the DNS features that are indicative of malicious behavior, we tracked and studied the DNS usage of several thousand well-known benign and malicious domains for a period of several months (we obtained these domains from the sources described in Section 4). After this analysis period, we identified 15 features that are able to characterize malicious DNS usage. Table1 gives an overview of the components of the DNS requests that we analyzed (i.e., feature sets) and the features that we identified. In the following sections, we describe these features and explain why we believe that they may be indicative of malicious behavior.

3.1 Time-Based Features

The first component of a DNS record that we analyze is the time at which the request is made. Clearly, the time of an individual request is not very useful by itself. However, when we analyze many requests to a particular domain over time, patterns indicative of malicious behavior may emerge. In particular, we examine the changes of the volume (i.e., number) of requests for a domain. The time-based features that we use in our analysis are novel and have not been studied before in previous approaches.

One of our insights is that malicious domains will often show a sudden increase followed by a sudden decrease in the number of requests. This is because malicious services often use a technique called *domain flux* [34] to make their infrastructures more robust and flexible against take downs. Each bot may use a domain generation algorithm (DGA) to compute a list of domains to be used as the command and control server or the dropzone. Obviously, all domains that are generated by a DGA have a short life span since they are used only for a limited duration. Examples of malware that make use of such DGAs are Kraken/Bobax [10], the Srizbi bots [41] and the Conficker worm [32]. Similarly, malicious domains that have recently been registered and are involved in scam campaigns will show an abrupt increase in the number of requests as more and more victims access the site in a short period of time.

To analyze the changes in the number of requests for a domain during a given period of time, we divide this period into fixed length intervals. Then, for each interval, we can count the number of DNS queries that are issued for the domain. In other words, the collection of DNS queries that target the domain under analysis can be converted into time series (i.e., chronologically ordered sequences of data values). Hence, we can leverage off-the-shelf algorithms [12]. We perform our time series analysis on two different scopes: First, we analyze the time series globally. That is, the start and end times of the time series are chosen to be the same as the start and the end times of the entire monitoring period. Second, we apply local scope time series analysis where the start times and end times are the first and last time the domain is queried during the analysis interval. While the global scope analysis is used for detecting domains that either have a short life or have changed their behavior for a short duration, the local scope analysis focuses on how do-

[1]We have experimentally determined the optimal training period to be seven days (see Section 4.2.)

Feature Set	#	Feature Name
Time-Based Features	1	Short life
	2	Daily similarity
	3	Repeating patterns
	4	Access ratio
DNS Answer-Based Features	5	Number of distinct IP addresses
	6	Number of distinct countries
	7	Number of domains share the IP with
	8	Reverse DNS query results
TTL Value-Based Features	9	Average TTL
	10	Standard Deviation of TTL
	11	Number of distinct TTL values
	12	Number of TTL change
	13	Percentage usage of specific TTL ranges
Domain Name-Based Features	14	% of numerical characters
	15	% of the length of the LMS

Table 1: Features.(LMS = Longest Meaningful Substring)

mains behave during their life time.

A domain is defined to be a *short-lived domain* (i.e., Feature 1) if it is queried only between time t_0 and t_1, and if this duration is comparably short (e.g., less than several days). A domain that suddenly appears in the global scope time series and disappears after a short period of activity has a fairly abnormal behavior for being classified as a benign domain. Normally, if a domain is benign, even if it is not very popular, our thesis is that the number of queries it receives should exceed the threshold at least several times during the monitoring period (i.e., two and a half months in our experiments). Therefore, its time series analysis will not result in an abrupt increase followed by a decrease as the time series produced by a short-lived domain does.

The main idea behind performing local scope analysis is to zoom into the life time of a domain and study its behavioral characteristics. We mainly focus on three features (i.e., Features 2, 3, 4) that may distinguish malicious and benign behavior either by themselves or when used in conjunction with other features. All the features involve finding similar patterns in the time series of a domain. Feature 2 checks if there are domains that show daily similarities in their request count change over time (i.e., an increase or decrease of the request count at the same intervals everyday). Feature 3 aims to detect regularly repeating patterns. Finally, Feature 4 checks whether the domain is generally in an "idle" state (i.e., the domain is not queried) or is accessed continuously (i.e., a popular domain).

The problem of detecting both short-lived domains and domains that have regularly repeating patterns can be treated as a change point detection (CPD) problem. CPD algorithms operate on time series and their goal is to find those points in time at which the data values change abruptly. The CPD algorithm that we implemented [12] outputs the points in time the change is detected and the average behavior for each duration. In the following section, we explain how we interpret the output of the CPD to detect the short-lived domains and the domains with regularly repeating patterns.

3.1.1 Detecting abrupt changes

As CPD algorithms require the input to be in a time series format, for each domain, we prepare a time series representation of their request count change over time. Our interval length for each sampling point is 3600 seconds (i.e., one hour). We chose 3600 seconds as the interval length after experimenting with different values (e.g., 150, 300 etc.).

Before feeding the input directly into the CPD algorithm, we normalize the data with respect to the local maximum. Then, we make use of the well-known CUSUM (cumulative sum) robust CPD algorithm that is known to deliver good results for many application areas [12]. CUSUM is an online algorithm that detects changes as soon as they occur. However, since we record the data to a database before analyzing it, our offline version of the CUSUM algorithm yields even more precise results (i.e., the algorithm knows in advance how the "future" traffic will look like as we have already recorded it).

Our algorithm to identify change points works as follows: First, we iterate over every time interval $t = 3600$ seconds, from the beginning to the end of the time series. For each interval t, we calculate the average request count P_t^- for the previous $\epsilon = 8$ time intervals and the traffic profile P_t^+ for the subsequent ϵ intervals. We chose ϵ to be

8 hours based on the insight that a typical day consists of three important periods: working time, evening and night. Second, we compute the distance $d(t)$ between P_t^- and P_t^+. More precisely:

$$P_t^- = \sum_{i=1}^{\epsilon} \frac{P_{t-i}}{\epsilon} \quad P_t^+ = \sum_{i=1}^{\epsilon} \frac{P_{t+i}}{\epsilon} \quad d(t) = \left| P_t^- - P_t^+ \right| \tag{1}$$

The ordered sequence of values $d(t)$ forms the input to the CUSUM algorithm. Intuitively, a change point is a time interval t for which $d(t)$ is sufficiently large and is a local maximum.

The CUSUM algorithm requires two parameters. The first parameter is an upper bound ($local\_max$) for the normal, expected deviation of the present (and future) traffic from the past. For each time interval t, CUSUM adds $d(t) - local\_max$ to a cumulative sum S. The second parameter determines the upper bound ($cusum\_max$) that S may reach before a change point is reported. To determine a suitable value for $local\_max$, we require that each individual traffic feature may deviate by at most $allowed\_avg\_dev = 0.1$. Based on this, we can calculate the corresponding value $local\_max = \sqrt{dim \times allowed\_avg\_dev^2}$. Since in our application, there is only one dimension, the $local\_max = allowed\_avg\_dev$. For $cusum\_max$, we use a value of 0.4. Note that we determined the values for $allowed\_avg\_dev$ and $cusum\_max$ based on empirical experiments and measurements.

The CPD algorithm outputs the average request count for each period a change is detected and the time that the change occurs. Since we employ the CPD algorithm for two purposes (namely to detect short-lived domains and domains that have repeating patterns), we run it twice. We first use the global scope time series and then the local scope time series as input. When the CPD is run with global time series, it can detect short-lived domains. Short-lived domains tend to have two sudden behavioral changes, whereas domains that are continuously queried have multiple change points. On the other hand, to detect the domains with repeating patterns on their local scope time series, we associate the number of the changes and the standard deviation of the durations of the detected changes.

3.1.2 Detecting similar daily behavior

A typical technique to measure the level of similarity of two time series is to calculate the distance between them [23]. To determine whether a domain produces similar time series every day, we calculate the Euclidean Distance between every pair of time series of a domain. Euclidean Distance is a popular distance measuring algorithm that is often used in data mining [14, 37, 43].

We first need to break the local time series produced for each domain into daily time series pieces. Each day starts at 00:00 am and finishes at 23:59 pm. Assuming that a domain has been queried n days during our analysis period, and $d_{i,j}$ is the Euclidean Distance between i_{th} day and j_{th} day, the final distance D is calculated as the average of $(n-1) * (n-2)/2$ different distance pairs, as shown in the following formula:

$$D = (\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{i,j})/((n-1) * (n-2)/2) \tag{2}$$

Using the Euclidean Distance, the results are sensitive to small variations in the measurements (e.g., 1000 requests between 9 and 10 am compared to 1002 requests between the same time period may fail to produce a correct similarity result although the difference is not significant). A common technique to increase the correctness of the results is to apply preprocessing algorithms to the time series before calculating the Euclidean Distance [17]. In our preprocessing step, we transform the time series $T = t_1, t_2, ..., t_n$, where n is number of intervals, into two phases. In the first phase, we perform offset translation by subtracting the mean of the series from each value (i.e., $T = T - mean(T)$). In the second phase, we scale the amplitude by dividing each value by the variance (i.e., $T = (T - mean(T))/std(T)$) [17].

3.2 DNS Answer-Based Features

The DNS answer that is returned by the server for a domain generally consists of several DNS A records (i.e., mappings from the host to IP addresses). Of course, a domain name can map to multiple IP addresses. In such cases, the DNS server cycles through the different IP addresses in a round robin fashion [1] and returns a different IP mapping each time. This technique is useful in practice for load balancing.

Malicious domains typically resolve to compromised computers that reside in different Autonomous Systems (ASNs), countries, and regions. The attackers are opportunistic, and do not usually target specific countries or IP ranges. Whenever a computer is compromised, it is added as an asset to the collection. Also, attackers typically use domains that map to multiple IP addresses, and IPs might be shared across different domains.

With this insight, we extracted four features from the DNS answer (i.e., feature set F2). The first feature is the number of different IP addresses that are resolved for a given domain during the experiment window (Feature 5). The second feature is the number of different countries that these IP addresses are located in (Feature 6). The third feature is the reverse DNS query results of the returned IP addresses (Feature 7). The fourth feature (Feature 8) is the

number of distinct domains that share the IP addresses that resolve to the given domain. Note that Features 5, 6, and 7 have been used in previous work (e.g., [**?**, 11, 30, 36]).

Although uncommon, benign domains may also share the same IP address with many other domains. For example, during our experiments, we saw that one of the IP addresses that belongs to *networksolutions.com* is shared by $10,837$ distinct domains. This behavior is sometimes exhibited by web hosting providers and shared hosting services.

To determine if an IP is used by a shared hosting service, we query Google with the reverse DNS answer of the given IP address. Legitimate web hosting providers and shared hosting services are typically ranked in the top 3 query answers that Google provides. This helps us reduce false positives.

3.3 TTL Value-Based Features

Every DNS record has a *Time To Live* (TTL) that specifies how long the corresponding response for a domain should be cached. It is recommended that the TTL is set to between 1 and 5 days so that both the DNS clients and the name servers can benefit from the effects of DNS caching [3].

Systems that aim for high availability often set the TTL values of host names to lower values and use Round-Robin DNS. That is, even if one of the IP addresses is not reachable at a given point in time, since the TTL value expires quickly, another IP address can be provided. A representative example for such systems are Content Delivery Networks (CDNs).

Unfortunately, setting lower TTL values and using Round-Robin DNS is useful for the attackers as well. Using this approach, malicious systems achieve higher availability and become more resistant against DNS blacklisting (DNSBL) [5] and take downs. For example, Fast-Flux Service Networks (FFSN) [36] are malicious systems that abuse Round-Robin DNS.

Most techniques to detect FFSNs are based on analyzing abnormal usage patterns of Round-Robin DNS. More precisely, to label a domain as being a member of an FFSN, previous research [30, 36] expects to observe a low TTL usage combined with a constantly growing DNS answers list (i.e., distinct IP addresses).

We extracted five features from the TTL value included in the DNS answers (see Table 1). The average TTL usage feature (Feature 9) was introduced in previous research [30]. The rest of the features (i.e., Features 10, 11, 12, 13) have not been used before in previous work.

During our experiments with large volumes of DNS traffic, we observed that frequent TTL changes are exhibited by malicious networks that have a sophisticated infrastructure. In such networks, some of the bots are selected to be proxies behind which other services (e.g., command and control servers) can be hidden. The managers of such malicious networks assign different levels of priorities to the proxy bots by setting lower TTL values to the hosts that are less reliable. For example, there is a good chance that a proxy running on an ADSL line would be less reliable than a proxy running on a server running in a university environment.

To determine the validity of our assumption about this type of TTL behavior, we tracked the Conficker domains for one week. We observed that different TTL values were returned for the IPs associated with the Conficker domains. While the static IP addresses have higher TTL values, the dynamic IP addresses, that are most probably assigned to home computers by Internet service providers, have lower TTL values (e.g., adsl2123-goland.net would have a lower TTL value than a compromised host with the domain name workstation.someuniversity.edu).

We observed that the number of TTL changes and the total number of different TTL values tend to be significantly higher in malicious domains than in benign domains. Also, malicious domains exhibit more scattered usage of TTL values. We saw that the percentage for the usage of some specific ranges of TTL values is often indicative of malicious behavior. Based on our empirical measurements and experimentations, the TTL ranges that we investigate are $[0, 1)$, $[1, 10)$, $[10, 100)$, $[100, 300)$, $[300, 900)$, $[900, inf)$. Malicious domains tend to set their TTL values to lower values compared to benign domains. In particular, the range of $[0, 100)$ exhibits a significant peak for malicious domains.

3.4 Domain Name-Based Features

Benign services usually try to choose domain names that can be easily remembered by users. For example, a bank called "The Iceland Bank" might have a domain name such as "www.icelandbank.com". In contrast, attackers are not concerned that their domain names are easy to remember. This is particularly true for domain names that are generated by a DGA.

The main purpose of DNS is to provide human-readable names to users as they often cannot memorize IP addresses of servers. Therefore, benign Internet services tend to choose easy-to-remember domain names. In contrast, having an easy-to-remember domain name is not a concern for people who perform malicious activity. This is particularly true in cases where the domain names are generated by a DGA. To detect such domains, we extracted two features from the domain name itself: First, the ratio of the numerical characters to the length of the domain name (Feature 14), and second, the ratio of the length of the longest meaningful substring (i.e., a word in a dictionary) to the length of the domain name (Feature 15).

Note that there exist popular domains such as *yahoo.com*

and *google.com* that do not necessarily include "meaningful" words. In order to gain a higher confidence about a domain, we query Google and check to see if it returns a hit-count for a domain that is above a pre-defined threshold.

When analyzing a domain, we only focus on the second level domains (i.e., SLD). For example, for x.y.server.com, we would take server.com. To detect domain names that have been possibly automatically generated, we calculate the percentage of numerical characters (Feature 14) and the ratio of the length of the longest meaningful substring to the total length of the SLD (Feature 15). To extract all possible meaningful substrings from an SLD, we check the English dictionary.

As some benign domains in China and Russia consist of combinations of alphabetical and numerical characters, Feature 15 produces a high positive rate. However, when Features 14 and 15 are combined, the false positives decrease. Also, for domains that are determined to be suspicious, we check how many times it is listed by Google. The reasoning here is that sites that are popular and benign will have higher hit counts.

4 Building Detection Models

4.1 Constructing the Training Set

The quality of the results produced by a machine learning algorithm strongly depends on the quality of the training set [35]. Our goal is to develop a classifier that is able to label domains as being benign, or malicious. Thus, we require a training set that contains a representative sample of benign and malicious domains. To this end, we studied several thousand malicious and benign domains, and used them for constructing our training set.

We collected malicious domains from multiple sources. Specifically, we obtained malicious domains from malwaredomains.com [19], the Zeus Block List [26], Malware Domains List [25], Anubis [13] reports, a list of domains that are extracted from suspected to be malicious URLs analyzed by Wepawet [18], and Phishtank [31]. We also used the list of domains that are generated by the DGAs of the Conficker [32] and Mebroot [34] (i.e., Torpig) botnets. These malicious domain lists represent a wide variety of malicious activity, including botnet command and control servers, drive-by download sites, phishing pages, and scam sites that can be found in spam mails.

Note that we are conservative when constructing the malicious domain list. That is, we apply a preliminary check before labeling a domain as being malicious and using it in our training set. Malicious domain sources such as Wepawet and Phishtank operate on URLs that have been submitted by users. Hence, while most URLs in these repositories are malicious, not all of them are. Also, while some third level domains (3LD) of a domain extracted from a URL may behave maliciously, the rest may not (e.g., a.x.com might be malicious, while x.com might be benign).

Assuming that a domain that is suspected to be malicious either by Wepawet or Phishtank has t_{total} possible 3LDs (number of distinct 3LD recorded by EXPOSURE during the analysis period) and t_{mal} 3LDs are thought to be malicious, we choose the domain to be representative for a malicious behavior only if t_{mal}/t_{total} is greater than 0.75 (i.e., only if 75% of the 3LDs have been reported to be involved in malicious activities). The initial malicious domain list that we generated consists of 3500 domains.

As discussed in detail in Section 5.1, we assume that all of the Alexa top 1000 domains and domains that we have observed on our sensors that are older than one year are benign. Therefore, we construct our initial benign domain list using these domains. However, to ensure that our benign domain list does not include any domain that might have been involved in malicious activity, we perform a two-step verification process.

First, we compare all the domains in the benign domain list with the malicious domain list and with the tools that test domains for their maliciousness, specifically with McAffee Site Advisor and Norton Safe Web. Second, we also cross-check the benign domain with the list provided by the Open Directory Project (ODP – a large, human-edited directory of the web constructed and maintained by volunteer editors). Our initial benign domain list consists of 3000 domains.

4.2 The Initial Period of Training

By experimenting with different values, we determined that the optimal period of initial training for our system was seven days. This period is mainly required for us to be able to use the time-based features that we described in Section 3. During this time, we can observe the time-based behavior of the domains that we monitor and can accurately take decisions on their maliciousness.

After the initial one week of training, we are able to retrain the system every day, hence, increasing detection accuracy.

4.3 The Classifier

Our classifier is built as a J48 decision tree algorithm (J48). J48 [40] is an implementation of the C4.5 algorithm [33] that is designed for generating either pruned or unpruned C4.5 decision trees. It constructs a decision tree from a set of labeled training set by using the concept of information entropy (i.e., the attribute values of the training set).

The J48 algorithm leverages the fact that the tree can be split into smaller subtrees with the information obtained

from the attribute values. Whenever the algorithm encounters a set of items that can clearly be separated from the other class by a specific attribute, it branches out a new leaf according to the value of the attribute. Each time a decision needs to be taken, the attribute with the highest normalized gain is chosen. Among all possible values of the attributes, if there is any value for which there is no ambiguity, the branch is terminated and the appropriate label is assigned to it. The splitting procedure stops when all instances in all subsets belong to the same class.

We use a decision tree classifier because these algorithms have shown to be efficient while producing accurate results [33]. As the decision tree classifier builds a tree during the training phase, the features that are best in separating the malicious and the benign domains can be clearly seen.

Recall that we divided the 15 features that we use into four different classes according to the type of information used: Features that are extracted from the time series analysis (F1, Time-Based Features), the DNS answer analysis (F2, DNS Answer-Based Features), the TTL value analysis (F3, TTL Value-Based Features), and the analysis of the domain name (F4, Domain Name-Based Features).

To find the combination of features that produce the minimum error rate, we trained classifiers using different combinations of feature sets and compared the results. Figure 2 shows the percentage of the number of miss-classified items with three different training schemes: 10-fold cross validation, 66% percentage split, and training on the whole training set. Note that the smallest error rates were produced by F1. Therefore, while experimenting with different combinations of feature sets, we excluded the combinations that do not include F1 (i.e., F23, F24, F34 and F234). The highest error rates are produced by F3 and F4. However, when all features are combined (i.e., F-all), the minimum error rate is produced. Hence, we use the combination of all the features in our system.

5 Evaluation

5.1 DNS Data Collection for Offline Experiments

Our sensors for the SIE DNS feeds receive a large volume of traffic (1 million queries per minute on average). Therefore, during our offline experimental period of two and a half months, we monitored approximately 100 billion DNS queries. Unfortunately, tracking, recording and post-processing this volume of traffic without applying any filtering was not feasible in practice. Hence, we reduced the volume of traffic that we wished to analyze to a more manageable size by using two filtering policies. The goal of these policies was to eliminate as many queries as possible that were not relevant for us. However, we also had to make sure that we did not miss relevant, malicious domains.

The first policy we used whitelisted popular, well-known domains that were very unlikely to be malicious. To create this whitelist, we used the Alexa Top 1000 Global Sites [4] list. Our premise was that the most popular 1000 websites on the Internet would not likely be associated with domains that were involved in malicious activity. These sites typically attract many users, and are well-maintained and monitored. Hence, a malicious popular domain cannot hide its malicious activities for long. Therefore, we did not record the queries targeting the domains in this whitelist. The domains in the whitelist received 20 billion queries during two and a half months. By applying this first filtering policy, we were able to reduce 20% of the traffic we were observing.

The second filtering policy targeted domains that were older than one year. The reasoning behind this policy was that many malicious domains are disclosed after a short period of activity, and are blacklisted. As a result, some miscreants have resorted to using domain generation algorithms (DGA) to make it more difficult for the authorities to blacklist their domains. For example, well-known botnets such as Mebroot [34] and Conficker [32] deploy such algorithms for connecting to their command and control servers. Typically, the domains that are generated by DGAs and registered by the attackers are new domains that are at most several months old. In our data set, we found 45.000 domains that were older than one year. These domains received 40 billion queries. Hence, the second filtering policy reduced 50% of the remaining traffic, and made it manageable in practice.

Clearly, filtering out domains that do not satisfy our age requirements could mean that we may miss malicious domains for the training that are older than one year. However, our premise is that if a domain is older than one year and has not been detected by any malware analysis tool, it is not likely that the domain serves malicious activity. To verify the correctness of our assumption, we checked if we had filtered out any domains that were suspected to be malicious by malware analysis tools such as Anubis and Wepawet. Furthermore, we also queried reports produced by Alexa [4], McAfee Site Advisor [8], Google Safe Browsing [6] and Norton Safe Web [9]. 40 out of the $45,000$ filtered out domains (i.e., only 0.09%) were reported by these external sources to be "risky" or "shady". We therefore believe that our filtering policy did not miss a significant number of malicious domains because of the pre-filtering we performed during the offline experiments.

5.2 Evaluation of the Classifier

To evaluate the accuracy of the J48 DecisionTree Classifier, we classified our training set with 10-fold cross-validation and percentage split, where 66% of the training set is used for training, and the rest is used to check the cor-

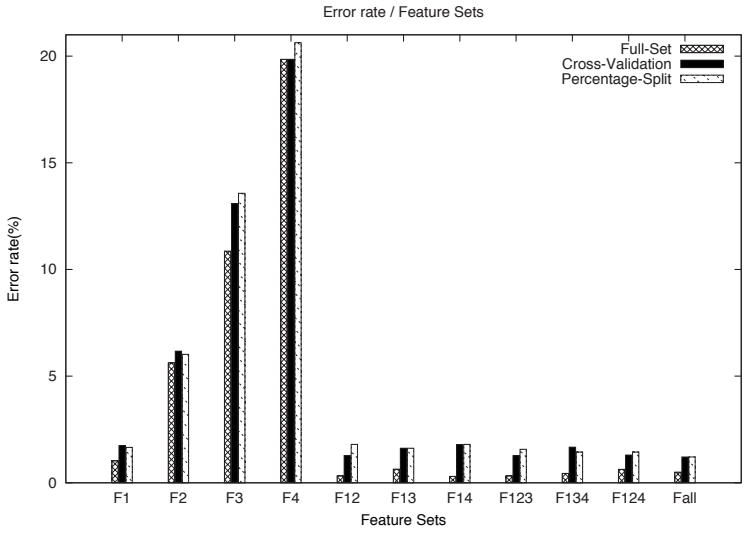


Figure 2: Percentage of miss-classified instances

rectness. Table 3 reports the results of the experiment. The Area Under the ROC curve [15] for the classifier is high for both methods.

Note that the false positive rate is low (i.e., around 1% for both methods). After investigating the reasons for the miss-classifications, we saw that the classifier had identified 8 benign domains as being malicious. The reason for the misclassification was that these domains were only requested a small number of times during the two and half months of experiments (i.e., making the classifier conclude that they were short-lived) and because they exhibited TTL behavior that looked anomalous (e.g., possibly because there was a configuration error, or because the site maintainers were experimenting to determine the optimal TTL value).

5.3 Experiments with the Recorded Data Set

During the two and a half month offline experimental period, we recorded and then analyzed 4.8 million distinct domain names that were queried by real Internet users. Note that a domain that only receives a few requests cannot produce a time series that can then be used for the time-based features we are analyzing. This is because a time series analysis produces accurate results only when the sampling count is high enough. In order to find the threshold for the minimum number of queries required for each domain, we trained our known malicious and benign domain list with differing threshold values. Figure 4 shows the detection and false positive rates for the threshold values we tested. Based on these empirical results, we set the threshold to 20 queries, and excluded the 4.5 million domains from our experiments that received less than 20 requests in the two and a half months duration of our monitoring.

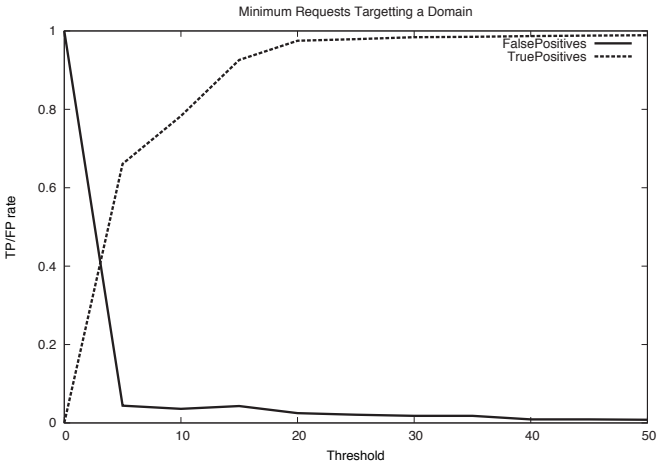


Figure 4: The effect of the minimum request count on detection rate

For further experiments, we then focused on the remaining 300,000 domains that were queried more than 20 times. EXPOSURE decided that 17,686 out of the 300,000 domains were malicious (5.9%).

5.3.1 Evaluation of the Detection Rate

The percentage split and cross-validation evaluations on the training set show that the detection rate of our classifier is around 98%. Since our goal is to be able to detect unknown malicious domains that have not been reported by any malicious domain analyzer, our evaluation of the classifier needs to show that we are able to detect malicious domains that do not exist in our training set. To this end, we used mal-

	AUC	Detection Rate	False Positives
Full data	0.999	99.5%	0.3%
10-folds Cross-Validation	0.987	98.5%	0.9%
66% Percentage Split	0.987	98.4%	1.1%

Figure 3: Classification accuracy. (AUC=Area Under the ROC Curve)

wareurls.com, a malware domains list that we had not used as a source for the initial malicious domains training set.

During the period we performed our experiments, malwareurls.com reported 569 domains as being malicious. Out of these 569 domains, 216 domains were queried by the infected machines in the networks that we were monitoring. The remaining 363 malware domains were not requested. Therefore, in our detection rate evaluation, we take into account only the 216 requested domains.

5 of the 216 domains were queried less than 20 times during entire monitoring period. Since we filter out domains that are requested less than 20 times, we only fed the remaining 211 domains to our system. In the experiments, all of these domains (that were previously unknown to us) were automatically detected as being malicious by EXPOSURE. Hence, the detection rate we observed was similar to the detection rate (i.e. 98%) estimated by the percentage split and cross-validation evaluations on the training set.

Obviously, our approach is not comprehensive and cannot detect all malicious domains on the Internet. However, its ability to detect a high number of unknown malicious domains from DNS traffic is a significant improvement over previous work.

5.3.2 Evaluation of the False Positives

As the domains in our data set are not labeled, determining the real false positive rate is a challenge. Unfortunately, manually checking all 17,686 domains that were identified as being malicious is not feasible. This is because it is difficult, in practice, to determine with certainty (in a limited amount of time) that a domain that is engaged in suspicious behavior is indeed malicious. Nevertheless, we conducted three experiments to make estimates about the false positives of our detection.

In order to obtain more information about the domains in our list, we first tried to automatically categorize them into different groups. For each domain, we started Google searches, checked well-known spamlists, and fed the domains into Norton Safe Web (i.e., Symantec provided us internal access to the information they were collecting about web pages). We divided the domains into ten groups: spam domains (Spam), black-listed domains (BlackList), malicious Fast-Flux domains (FastFlux), domains that are queried by malware that are analyzed by malware analysis tools (Malware), Conficker domains (Conficker), domains that have adult content, domains that are suspected to be risky by Norton Safe Web and McAfee Site Advisor (Risky), phishing domains (Phishing), domains about which we were not able to get any information either from Google or from other sources (No Info), and finally, benign domains that are detected to be malicious (False Positives) (See Table 2).

In the first experiment, we manually investigated 50 random malicious domains from our list of 17,686. We queried Google, checked websites that discuss malicious networks, and tried to identify web links that reported a malicious behavior by the domain. Among the 50 randomly chosen domains, the classifier detected three benign domains as being malicious. All these domains had an abnormal TTL change behavior.

In the second experiment, we automatically cross-checked the malicious and suspicious domains that we had identified with our classifier using online site rating tools such as McAfee Site Advisor, [8], Google Safe Browsing [6] and Norton Safe Web [9]. The results show that the false positive estimate is around 7.9% for the malicious domains that we identified. Given that our classifier is able to identify malicious domains automatically, these false positive rates are acceptable for the tool to be deployed a large-scale, early warning system.

Note that EXPOSURE did not generate any false positives during the two week real-time, real-world deployment in an ISP as discussed in the next section.

5.4 Real-World, Real-Time Detection with EXPOSURE

To test the feasibility and scalability of EXPOSURE as a malicious domain detector in real-life, we deployed it in the network of an ISP that provided us complete access to its DNS servers for two weeks. These servers receive DNS queries from a network that supports approximately 30,000 clients.

During the two-week experimental period, EXPOSURE analyzed and classified 100 million DNS queries. No pre-filtering was applied. At the end of two weeks, EXPOSURE detected 3117 new malicious domains that were previously not known to the system and had not been used

MW-Group	Rand 50	Malicious
Spam	18	3691
Black-List	8	1734
FastFlux	-	114
Malware	6	979
Conficker	4	3693

MW-Group	Rand 50	Malicious
Adult	3	1716
Risky	-	788
Phishing	3	0
No Info	5	2854
False Positives	3 (6%)	1408 (7.9%)

Table 2: Tests for False Positives

in the training. 2821 of these domains fall into the category of domains that are generated by a DGA and all belong to the same malicious entity. 5 out of the remaining 396 domains were reported as being malicious domains by security companies such as Anvira, one month after we had detected them.

We cross-checked the rest of the remaining domains we had detected. All detected domains were classified as being risky by McAfee Site Advisor [8].

Figures 5(a) and 5(b) show the number of new, previously unknown malicious domains detected every day. As can be seen, after the initial seven days of local training in the network being monitored, EXPOSURE started to produce daily detections and detected 200 new malicious domains per day on average.

After the experiments, we provided the ISP with the list of clients that were potentially infected, or had been victims of scams.

The distinct number of IP addresses that queried the malicious domains that EXPOSURE detected were 3451. Since the ISP applies a dynamic IP assignment to its clients, this number does not represent the exact number of infected machines in the network. To estimate the number of infected machines in the network, we grouped the malicious domains according to the IP addresses they are mapped to. There were 5 different groups of malicious domains. We then calculated the average number of distinct IP addresses that issued DNS queries to the domains in these 5 groups every hour. We chose one hour as an interval by assuming that the users in the network stay online at least an hour before they disconnect. Table 3 lists the number of clients that attempt to access the domains that fall into the different malware groups. We estimate that there were about 800 machines on the network that issued the requests to the malicious domains.

5.5 Comparison with Previous Work

5.5.1 The Fast-Flux Detectors

The results in Table 2 show that 114 of the malicious domains that our classifier has identified during the initial training phase fall in to category of Fast-Flux Service Networks (FFSNs). Since we claim that our approach is able to detect a wide range of malicious domains including FFSNs, we compare our detection rate for this threat with the most recent published work in this area (i.e., [30]).

Perdisci et. al. [30] filters out all of the domains that are not likely to be classified as being FFSN. When we applied the same policy to our two and half month data set, 300,000 domains were filtered out and 5,771 were left as candidates for FFSNs. When we classified these domains with the feature set Perdisci et. al. use in their paper, we detected 114 FFSNs using their approach. Hence, our approach is as good in detecting FFNSs as Perdisci et. al. although it is a much more generic system.

5.5.2 Notos: Reputation-based Malicious Domain Detection

Very recently, Antonakakis et al. [11] concurrently and independently proposed a detection scheme that is similar to our work. The proposed system, Notos, dynamically assigns reputation scores to domain names whose maliciousness has not been discovered yet. A detection scheme is built that is based on the premise that agile malicious uses of DNS have unique characteristics. Hence, the claim is that malicious use of DNS can be distinguished from benign use.

To be able to define these unique characteristics, the authors analyze a number of features that are grouped into three categories: Network-based features, zone-based features (i.e. features that are extracted from the domain name itself, either by string analysis or with the information obtained from whois service) and evidence-based features.

While the network-based features are employed for combing out the domains that do not exhibit fluxy behavior (i.e. stable DNS usage), the zone-based features are used for distinguishing between legitimate CDNs and the domains that are likely to be malicious. After this two-layer classification, reputation scores are given to the domains. In other words, all of the domains and the IP addresses they are mapped to are compared with already known lists of domains or IP addresses that host malicious entities. This third step of classification is done using evidence-base features.

Groups	Avg Life Time	Most frequent life time	# of infected clients
DGA domains	1.2 days	0.99 days	49
Iksmas Worm	11.9 days	11.9 days	70
Worm:Win32/Slenping	12.0 days	12.0 days	253
Trojan-Generic.dx	11.9 days	11.9 days	70
Other	10.8 days	11.9 days	391
Total			833

Table 3: Information on the detected malicious domains

In their paper, as a limitation of Notos, the authors state that Notos is not able to detect malicious domains that are mapped to a new address space each time and never used for other malicious purposes again. This limitation stems from the fact that Notos strongly relies on network-based features. EXPOSURE does not have this limitation as it uses time-based features. Since such domains would have a short life, they would appear in the time series and disappear immediately after they are deactivated by the attacker. Hence, unlike Notos, we are able to detect such domains.

As discussed before, the 2821 automatically generated malicious domains that we detected in the real-life traffic of the ISP had an average lifespan of 1.2 days (see Table 3). That is, all these domains were short-lived domains. During their life time, on average, they mapped to 3.14 distinct IP addresses. In the first phase of Notos' detection scheme, the domains are divided into two categories: domains that have a stable network-model and domains that have a non-stable network-model. Since the automatically generated malicious domains that we are able to detect do not use a wide range of IP addresses, Notos might classify these domains as domains with a stable network profile. In other words, we believe that Notos might miss-classify them.

Also, as the authors discuss in their paper, because Notos is a reputation-based system, there may be cases where legitimate domains that are hosted in "bad neighborhoods" may be identified as being malicious. In comparison, the features that EXPOSURE relies on do not cause such false positives as no historical information on IPs or domains are utilized.

One main advantage of EXPOSURE over Notos is that Notos requires a large passive DNS collection and sufficient time to create an accurate, passive DNS database. First, it is unclear how much time is required for this database to be comprehensive. Second, this database needs to be constantly updated with large data-feeds in order to remain accurate and to have a wide overview of malicious activities on the Internet. In comparison, as we show in our evaluation, EXPOSURE only required a week of local training, and much less DNS data in the network of a medium ISP to be able to detect unknown domains.

6 Related Work

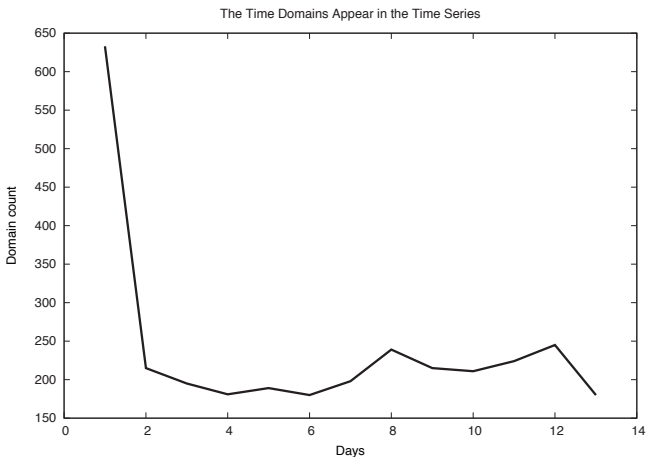
The Domain Name System (DNS) has been increasingly being used by attackers to maintain and manage their malicious infrastructures. As a result, recent research on botnet detection has proposed number of approaches that leverage the distinguishing features between malicious and benign DNS usage.

The first study [39] in this direction proposed to collect real-world DNS data for analyzing malicious behavior. The results of the passive DNS analysis showed that malicious domains that are used in Fast-Flux networks exhibit behavior that is different than benign domains. Similarly, Zdrnja et al. [42] performed passive monitoring to identify DNS anomalies. In their paper, although they discuss the possibility of distinguishing abnormal DNS behavior from benign DNS behavior, the authors do not define DNS features that can be used to do so.

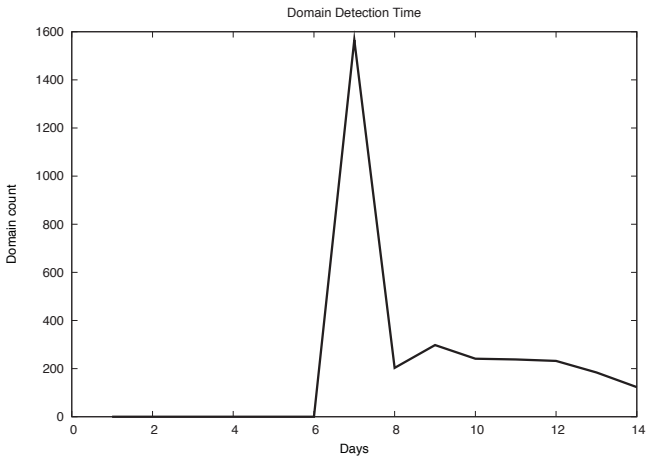
In general, botnet detection through DNS analysis follows two lines of research: The first line of research tries to detect domains that are involved in malicious activities. The goal is to identify infected hosts by monitoring the DNS traffic. The second line of research focuses on the behaviors of groups of machines in order to determine if they are infected (e.g., a collection of computers always contact the same domain repeatedly).

6.1 Identifying Malicious Domains

To detect malicious domains, previous approaches make use of passive DNS analysis, active DNS probing, and WHOIS [2] information. For example, recent work by Perdisci et al. [30] performs passive DNS analysis on recursive DNS traffic collected from number a number of ISP networks with the aim of detecting malicious Fast-Flux services. Contrary to the previous work [24, 28, 29, 36], Perdisci's work does not rely on analyzing blacklisted domains, and domains that are extracted from spam mails. Our work significantly distinguishes itself from theirs as we are able to detect all different kinds of malicious domains such as phishing sites, spamming domains, dropzones, and bot-



(a)



(b)

Figure 5: The first time a domain is queried and the first time it is detected

net command and control servers. We do not only focus on detecting Fast-Flux service networks.

A second branch of study that aims to detect malicious domains [27, 36] leverages active DNS probing methods. That is, the domains that are advertised to be malicious by various sources (e.g. spam mails) are repeatedly queried to detect the abnormal behavior. The main drawback of active DNS analysis is the possibility of being detected by the miscreants who manage the domains under analysis. Passive DNS analysis, in comparison, is more stealthy because of its non-intrusiveness characteristics.

Based on URL features they extract from spam mails, Ma et. al. [27] study a number of statistical methods for machine learning for classifying websites. In particular, they analyze spam URLs according to their lexical construction, and the information contained in the host name part of the URL. To obtain the information from the host name, they perform active probing to determine the number of IP addresses associated with the domain. Once they obtain the IP address list, they analyze the location of the IP address and to which ANS it belongs to. The main limitation of this system is that it performs the analysis only based on the domains that are included in spam mails. Hence, the system cannot see other classes of malicious domains such as command and control servers.

Another type of study on detecting malicious domains leverages properties inherent to domain registrations and their appearance in DNS zone files [20]. That is, they associate the registration information and DNS zone properties of domains with the properties of known blacklisted domains for proactive domain blacklisting. This method completely relies on historical information. Therefore, it is not able to detect domains that do not have any registration information and DNS zone commonalities with known blacklisted domains. On the other hand, our work, which does not require any historical information, is able to detect such domains.

6.2 Identifying Infected Machines by Monitoring Their DNS Activities

In [16], the authors propose an anomaly-based botnet detection mechanisms by monitoring group activities in the DNS traffic of a local network. The authors claim that there exist distinguishing features to differentiate DNS traffic generated by botnets and benign clients. Similarly, [38] also attempts to identify botnet DNS access behavior in a local network. The authors use a bayesian algorithm. In comparison to these existing works, we aim to identify malicious domains from DNS traffic in general, and do not only focus on botnets.

6.3 Generic Identification of Malicious Domains Using Passive DNS Monitoring

To date, only one system has been proposed that aims to detect malicious domains using passive DNS analysis. In a concurrent and independent work very recently presented by Antonakakis et al. [11], the authors present Notos. Notos dynamically assigns reputation scores to domain names whose maliciousness has not been discovered yet.

We have compared EXPOSURE with Notos in Section 5.5.2. EXPOSURE eliminates several shortcomings of Notos. It does not require a wide overview of malicious activities on the Internet, a much shorter training time, and is able to classify domains that Notos would miss-classify.

7 Limitations

A determined attacker who knows how EXPOSURE works and who is informed about the features we are looking for in DNS traffic might try to evade detection. To evade EXPOSURE, the attackers could try to avoid the specific features and behavior that we are looking for in DNS traffic. For example, an attacker could decide to assign uniform TTL values across all compromised machines. However, this would mean that the attackers would not be able to distinguish between more reliable, and less reliable hosts anymore and would take a reliability hit on their malicious infrastructures. As another example, the attackers could try to reduce the number of DNS lookups for a malicious domain so that only a single lookup is performed every hour (i.e., so that the malicious domain is blacklisted). However, this is not trivial to implement, reduces the attack's impact, and requires a high degree of coordination on the attacker's side. Even though it is possible for an attacker to stay below our detection radar by avoiding the use of these features, we believe that this comes with a cost for the attacker. Hence, our systems helps increase the difficulty bar for the attackers, forces them to abandon the use of features that are useful for them in practice, and makes it more complex for them to manage their infrastructures.

Clearly, our detection rate also depends on the training set. We do not train for the family of malicious domains that constitute attacks that are conceptually unknown and have not been encountered before in the wild by malware analyzers, tools, or experts. However, the more malicious domains are fed to the system, the more comprehensive our approach becomes over time.

Note that if the networks that we are monitoring and training our system on are not infected, obviously, we will not see any malicious domains. We believe that we can improve our ability to see more malicious attacks by having access to larger networks and having more installations of EXPOSURE.

8 Conclusions

The domain service (DNS) is a crucial component of the Internet. DNS provides a two-way mapping between domain names and their IP addresses. Just as DNS is a critical service for the functioning of benign Internet services, it has also started to play an important role for malicious activities. For example, bots resolve DNS names to locate their command and control servers, and spam mails contain URLs that link to domains that resolve to scam servers.

In this paper, we introduced EXPOSURE, a system that employs passive DNS analysis techniques to detect malicious domains. Our thesis is that it is beneficial to monitor the use of the DNS system on a large-scale for signs that indicate that a certain name is used as part of a malicious operation. Our experimental results show that our approach works well in practice, and that it is useful in automatically identifying a wide category of malicious domains such as botnet command and control servers, phishing sites, and scam hosts. Compared to related work, our approach is generic, and does only focus on a specific class of threat (e.g., such as Fast-Flux botnets).

We believe that EXPOSURE is a useful system that can help security experts and organizations in their fight against cyber-crime. As future work, we plan to release EXPOSURE to the public as a community service.

9 Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 257007. This work has also been supported in part by Secure Business Austria, the European Commission through project IST-216026-WOMBAT funded under the 7th framework program, by the ONR under grant N000140911042 and by the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537. We thank the Internet Security Consortium Security Information Exchange project (ISC@SIE) for providing portion of the DNS data used in our experiments. Additionally, we thank the University of Bergamo for providing access to their DNS traffic during the early phases of this work.

References

[1] RFC 1794 - DNS Support for Load Balancing. `http://tools.ietf.org/html/rfc1794`, 1995.

[2] RFC1834 - Whois and Network Information Lookup Service, Whois++. `http://www.faqs.org/rfcs/rfc1834.html`, 1995.

[3] RFC 1912 - Common DNS Operational and Configuration Errors. `http://www.faqs.org/rfcs/rfc1912.html`, 1996.

[4] Alexa Web Information Company. `http://www.alexa.com/topsites/`, 2009.

[5] DNSBL - Spam Database Lookup. `http://www.dnsbl.info/`, 2010.

[6] Google Safe Browsing. `http://www.google.com/tools/firefox/safebrowsing/`, 2010.

[7] Internet Systems Consortium. `https://sie.isc.org/`, 2010.

[8] McAfee SiteAdvisor. http://www.siteadvisor.com/, 2010.

[9] Norton Safe Web. http://safeweb.norton.com/, 2010.

[10] B. Amini. Kraken Botnet Infiltration. http://dvlabs.tippingpoint.com/blog/2008/04/28/kraken-botnet-infiltration, 2008.

[11] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a Dynamic Reputation System for DNS. In *19th Usenix Security Symposium*, 2010.

[12] Michle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, 1993.

[13] Ulrich Bayer, Christopher Kruegel, and Engin Kirda. TTAnalyze: A Tool for Analyzing Malware. In *15th EICAR Conference, Hamburg, Germany*, 2006.

[14] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, 2002.

[15] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. In *Pattern Recognition*, volume 30, pages 1145–1159, 1997.

[16] H. Choi, H. Lee, and H. Kim. Botnet detection by monitoring group activities in DNS Traffic. In *7th IEEE International Conference on Computer and Information Technologies*, 2007.

[17] Selina Chu, Eamonn Keogh, David Hart, Michael Pazzani, and Michael. Iterative deepening dynamic time warping for time series. In *In Proc 2 nd SIAM International Conference on Data Mining*, 2002.

[18] M. Cova. Wepawet. http://wepawet.iseclab.org/.

[19] Malware Domains. Malware Domain Block List. http://www.malwaredomains.com/, 2009.

[20] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *Proceedings of the Third USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET)*, San Jose, CA, USA, April 2010.

[21] F. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *10th European Symposium On Research In Computer Security*, 2005.

[22] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale Botnet Detection and Characterization. In *Usenix Workshop on Hot Topics in Understanding Botnets*, 2007.

[23] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *ACM SIGMOD Conference on Management of Data*, pages 151–162, 2001.

[24] M. Konte, N. Feamster, and J. Jung. Dynamics of online scam hosting infrastructure. In *In Passive and Active Measurement Conference*, 2009.

[25] Malware Domains List. Malware Domains List. http://www.malwaredomainlist.com/mdl.php, 2009.

[26] Zeus Block List. Zeus domain blocklist. https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist, 2009.

[27] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious urls. In *Proceedingsof theSIGKDD Conference. Paris,France*, 2009.

[28] J. Nazario and T. Holz. As the net churns: Fast-flux botnet observations. In *International Conference on Malicious and Unwanted Software*, 2008.

[29] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. Fluxor: Detecting and monitoring fast-flux service networks. In *Detection of Intrusions and Malware, and Vunerability Assessment*, 2008.

[30] R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. In *25th Annual Computer Security Applications Conference (ACSAC)*, 2009.

[31] Phishtank. Phishtank. http://www.phishtank.com/, 2009.

[32] P. Porras, H. Saidi, and V. Yegneswaran. A Foray into Conficker's Logic and Rendezvous Points. In *In USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.

[33] J.R. Quinlan. Learning with continuous classes. *Proceedings of the 5th Australian joint Conference on Artificial Intelligence*, Singapore: World Scientific:343 – 348, 1995.

[34] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *ACM Conference on Computer and Communication Security (CCS)*, 2009.

[35] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 2009.

[36] T.Holz, C. Gorecki, K. Rieck, and F.C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2008.

[37] D. Turaga, M. Vlachos, and O. Verscheure. On K-Means Cluster Preservation using Quantization Schemes. In *IEEE International Conference on Data Mining, ICDM09*, 2009.

[38] R. Villamarn-Salomn and J. C. Brustoloni. Bayesian bot detection based on DNS traffic similarity. In *SAC'09: ACM symposium on Applied Computing*, 2009.

[39] F. Weimer. Passive DNS Replication. In *FIRST Conference on Computer Security Incident*, 2005.

[40] IH. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[41] J. Wolf. Technical details of Srizbis domain generation algorithm. `http://tinyurl.com/6mdasc`, 2008.

[42] B. Zdrnja, N. Brownlee, and D. Wessels. Passive Monitoring of DNS anomalies. In *DIMVA*, 2007.

[43] H. Zitouni, S. Sevil, D. Ozkan, and P. Duygulu. Reranking of Image Search Results using a Graph Algorithm. In *9th International Conference on Pattern Recognition*, 2008.