# Winning with DNS Failures: Strategies for Faster Botnet Detection

Sandeep Yadav and A.L. Narasimha Reddy

Department of Electrical and Computer Engineering, Texas A&M University
sandeepy@tamu.edu,reddy@ece.tamu.edu

**Abstract.** *Botnets such as Conficker and Torpig utilize high entropy domains for fluxing and evasion. Bots may query a large number of domains, some of which may fail. In this paper, we present techniques where the failed domain queries (NXDOMAIN) may be utilized for: (i) Speeding up the present detection strategies which rely only on successful DNS domains. (ii) Detecting Command and Control (C&C) server addresses through features such as temporal correlation and information entropy of both successful and failed domains. We apply our technique to a Tier-1 ISP dataset obtained from South Asia, and a campus DNS trace, and thus validate our methods by detecting Conficker botnet IPs and other anomalies with a false positive rate as low as 0.02%. Our technique can be applied at the edge of an autonomous system for real-time detection.*

**KEY WORDS**
Botnet, Domain-fluxing, DNS, Failures

## 1 Introduction

Botnets have been used for spamming, phishing, DDoS (Distributed Denial of Service) attacks. Some botnets such as Kraken/Bobax, Torpig [8], and Conficker [6] utilize *fluxing* techniques, where the domain name of a C&C server changes rapidly (*domain fluxing*) or the IP address for a domain name is altered (*IP fluxing*). To automate the domain name generation for fluxing, botnet owners rely on generating domain names algorithmically. The domain names thus formed, comprise of alphanumeric characters chosen randomly, and which thus exhibit high *information entropy*. As the domain names for the C&C servers are short lived, and as only a fraction of this large set of domains may be used for actual DNS use, blacklisting techniques prove ineffective in countering such fluxing botnets.

Reverse engineering of bot executables may yield the domain name generation algorithm and subsequently the domain names that a bot may query in the future. These domain names may be blacklisted or pre-registered in advance by security researchers. Domain fluxing botnets overcome this vulnerability by choosing to generate a large number of names, where only a few of them may host the C&C server. The large number of domain names is expected to overwhelm the pre-registration by others and potentially provide a cover for the actual name of the C&C server used by the botnet.

Botnets that employ domain fluxing can be characterized by the following two important features: (a) The alphanumeric distribution or entropy of the domain names for C&C servers is considerably different from human generated names. (b) The bots generate many failed DNS queries as many of the algorithmically generated domain names may not be registered or not available as C&C servers. We exploit these two important properties to detect botnets with very low latency, where we define latency as the number of domain names required for successful anomaly detection (or the time taken to collect those domains).

With our approach, we analyze successful DNS queries, and the failed DNS queries within the vicinity of the successful queries, thus exploiting their features to not only detect the C&C servers of those botnets faster, but also simultaneously detect bots within the network. While our detection mechanism is designed specifically to detect domain fluxing botnets by utilizing DNS failures, previous approaches relying only on domain entropy analysis can still be used in the event that there are no DNS failures. By analyzing the failed queries along with the successful queries, we increase the data available for analysis and hence speed up the detection process. While our technique can be used online (or in real-time), we focus on a

This work is supported in part by a Qatar National Research Foundation grant, Qatar Telecom, and NSF grants 0702012 and 0621410.

trace-driven evaluation methodology here to keep the explanation simpler. Additionally, our analysis is based only on DNS network traffic, thereby reducing resource requirements, in comparison to techniques relying on general network traffic analysis.

When individual clients/hosts query a resolver, the failed queries can potentially be attributed to the presence of bots on that client and the successful queries close to the failures can be assumed to be related with high confidence. However, when queries are forwarded to a resolver from another local resolver or a DNS query aggregator, the queries from many clients can be grouped together and relating failed queries to other successful queries in the query stream becomes problematic. Our approach is cognizant of this difficulty and is capable of producing accurate results in the presence of aggregated query streams.

The main contributions of this work are:

- We utilize the failures around successful DNS queries and the entropy of the domains belonging to such queries, for *detecting* botnets with lower latency compared to previous techniques.
- We propose and evaluate a *speeding* technique which correlates DNS domain query failures for faster detection of domain fluxing botnets' C&C server IPs. We utilize temporal correlation between DNS queries and entropy-based correlation between domain names, for speedier detection.
- We show through a trace driven analysis that the proposed techniques can considerably speed up the detection of botnets that generate many DNS query failures. This in turn will constrain the domain name generation algorithms further if they want to evade detection by techniques such as proposed here.

We apply our techniques to two datasets. The first is a Tier-1 ISP dataset obtained from South Asia, captured for a period of approximately one day. Additionally, we analyze a university campus DNS trace captured over a month. The datasets consist of botnets validated through previous techniques applied to the trace [11]. Based on our analysis, we detect the presence of the recently discovered *Conficker* botnet. Our experiments indicate a false positive rate as low as 0.02% with a high detection rate. Our evaluation also yields how different features characterizing botnets can be varied, to assist a network administrator in tuning these parameters for their network(s).

The rest of this paper is organized as follows. Section 2 discusses the related work on botnet detection. In section 3, we discuss our methodology for *detection* of domain-fluxing botnets, as well as propose an alternate correlation criteria for *speeding* the state-of-the-art detection techniques. The results have been outlined in section 4 followed by the discussion on the limitations and security loopholes that attackers may exploit, in section 5. Finally, we draw the conclusions and highlight the future work in section 6.

## 2 Related Work

Alphabet entropy measures to detect algorithmically generated botnets by using successful domain name queries mapping to IP addresses, are proposed in [11]. Jiang et. al. [4] use DNS failures to determine suspicious activity within the local autonomous system. Their technique analyzes bipartite graphs between failed DNS domain names and querying clients, to determine connected components with anomalous activity. Our approach, additionally, analyzes related successful queries and detects the botnet C&C servers along with the bots. The authors in [13] analyze unproductive network traffic of multiple protocols to classify malicious hosts based on features such as rate of failed traffic generation, entropy of ports used etc. In our work, we do not require training of data, and only rely on DNS based features for botnet detection.

Botnet identification using DNS has been explored in [9] where the authors utilize query rates based features of successful and failed DNS queries, to identify botnet anomalies. [10] detects new bots based on the similarity in querying behavior for known malicious hosts. Our technique does not rely on query rates and can detect botnets even if each bot queries for an independent botnet C&C server's domain names. Previous work on botnet detection has also examined the correlation of network activity between time and space as exhibited by users within a network [3]. We, however, use only the DNS traffic for detecting botnet activity, drastically reducing the resource requirements. Also, DNS security has been investigated in a number of recent studies [12] which have focused on DNS indirection and cache poisoning prevention.

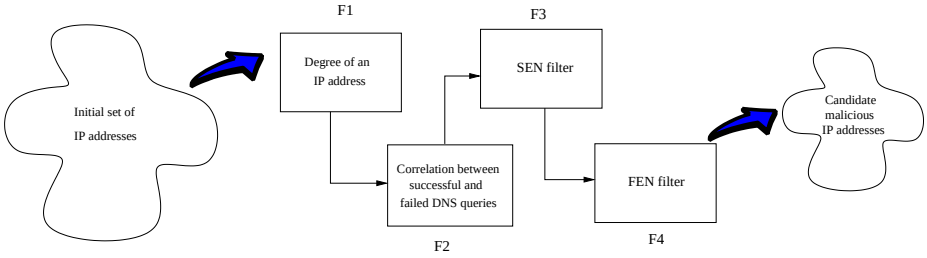



**Fig. 1.** Filtering steps

## 3 Methodology

In this section, we describe our technique for detecting botnets through DNS traffic analysis using successful and failed DNS queries. We also highlight correlating failed domains, for speedier detection of malicious IP addresses. Our primary goal is to detect the IP address of a domain fluxing botnet's C&C server. As a consequence of our analysis, we obtain the bots within the local network, and the domain names belonging to the botnet, thus exposing the botnet altogether. To discover anomalous IP address(es), we exploit multiple features such as the botnet structure and the temporal correlation between DNS query patterns of participating bots. Prior to analyzing the DNS traffic, we use a white-list for filtering out known benign DNS queries (details in section 4), leveraging a more accurate analysis.

### 3.1 Filtering steps

Figure 1 demonstrates the steps involved in narrowing down the set of IP addresses that are returned in DNS response packets (post white-listing), to a relatively smaller list of anomalous IP addresses. With each filter, we select a fraction of the input supplied by the previous filter, reducing the subsequent work. In the following subsections, we describe each filter applied for a candidate C&C IP address (denoted by $cncip$) resulting in $cncip$ being discarded as legitimate or subject to additional filters. The measures employed by each filter, are either computed using select or all the time windows in which the candidate IP address occurs. The typical time bin/window length used in our trace is subjective to the dataset in consideration. For evaluation presented in section 4, we typically use a 128 sec window (64 sec symmetric about $cncip$). The following subsections detail how each measure is computed.

**Degree of an IP address ($D_{cncip}$):** Domain fluxing is characterized by multiple domain names mapping to an IP address. We define the degree of an IP address as the number of domain names that map to a $cncip$. As a first filter, we use the degree ($F_1$ in Figure 1) to separate a set of IP addresses more likely to exhibit botnet like domain fluxing. For a given IP address, this number may vary based on the length of the trace analyzed. For instance, an IP address analyzed for an hour may have five domain names mapping to it. However, if analyzed for two hours, eight domain names may map to it, which includes previously expired domain names. While we consider the IPs which have a degree of at least two, we vary $D_{cncip}$ to evaluate how quickly we detect anomalies. For a typical analysis, we use a degree threshold of eight, independent of the time for which a candidate IP address is analyzed. Thus, the filter $F_1$ can be bypassed if an IP has less than eight domain names mapping to it. However, this puts a constraint on the fluxing that a botnet server can exhibit. It should be noted that, Content Distribution Networks (CDNs) also have a high degree. However, CDNs get separated through additional filters as described ahead.

**Correlation metric ($Corr_{cncip}$):** As introduced earlier, bots generate burst of DNS queries, a fraction of which may fail. Thus, we exploit the temporal correlation between DNS successes and failures to identify malicious behavior. On observing a time window of DNS queries for a bot, we may observe the presence of failures, more frequently, than for legitimate clients. It is represented by filter $F_2$ in Figure 1.

The correlation metric ($Corr_{cncip}$) for a candidate IP address is computed as the probability of observing at least one failed DNS query in a time bin, given that $cncip$ was returned as an answer to a successful

DNS query in the same bin. For detection, we heuristically choose the threshold as 0.5 implying that majority of windows in which a $cncip$ appears, should also have failures for it to be considered a meaningful anomaly. In section 4, we study how the false positives decrease on increasing this threshold or when the correlation metric changes upon restricting our analysis to windows with more failures.

We use the following equation to compute this metric:

$$Corr_{cncip} = \frac{\sum \text{Time bins with } (S_{cncip} \cap F_{client})}{\text{Time bins with } S_{cncip}} \quad (1)$$

where $S_{cncip}$ denotes the boolean condition of whether $cncip$ occurs in a time bin. $F_{client}$ refers to the boolean variable indicating the presence of at least one failure in the corresponding time bin for the *client*. The correlation metric is computed with the time series of all clients which receive $cncip$ as the DNS response address. This metric may not be sufficient in topologies comprising of DNS aggregators where the temporal co-occurrence of DNS failures and successes is more frequent. Further developed measures limit the errors produced due to DNS aggregators.

**Succeeding domain set entropy ($SEN_{cncip}$):** We use *edit distance* as a metric for determining the similarity between a pair of domain names. Algorithmically generated domains exhibit a high value for this metric, owing to limited similarity between a given pair of domains. However, domain names observed for a legitimate entity, frequently have repeated occurrence of certain characters, which lower the computed normalized edit distance (or the entropy associated with the entity), as substantiated by [11]. For instance, a pair of domain names such as *www.google.com, ns.google.com* have a lower normalized edit distance than a pair like *jswrts.ws, yvqcbtvztpm.cc*, as observed for Conficker.

Edit distance is defined as an integral value indicating the number of transformations required to convert a given string to the other. The type of eligible transformations include addition, deletion, and modification of a character. We use the normalized edit distance measure computed as the Levenshtein edit distance [5] between a pair of strings normalized by the length of the longer string. The entropy of domains mapping to an IP address (and hence successful DNS queries), $SEN_{cncip}$, is determined by computing the normalized edit distance between every pair of domains that map to $cncip$ (taken from set with cardinality $|D_{cncip}|$), and averaged over all such pairs. Therefore, the complexity of entropy calculation is $O(n^2)$ where $n$ is the number of domain names successfully mapping to an IP address over the duration of analysis. This duration is defined either in terms of a pre-determined time, or the first few successful domains encountered for a given $cncip$. Once $SEN_{cncip}$ is computed, if it exceeds a threshold (reserved for highly domain fluxing entities), we consider it for further analysis. Our evaluation shows that while high $SEN_{cncip}$ IPs may be detected easily, even enitities with relatively low entropy are detected with small false positive rates, making it difficult for botnet owners to improve their domain generation algorithm (DGA).

**Failing domain set entropy ($FEN_{cncip}$):** For a botnet, the domain name generation algorithm for failed domain names is no different than the domain names successfully resolved. The features expressed through alphanumeric characters composing the failed and successful DNS queries generated by a botnet, are therefore very similar. Thus, the failed domain names can help reduce the latency of analysis and improve detection since many more names can be analyzed in a shorter period of time, when associated with the succeeding queries.

To compute the entropy of failed domain names (denoted as $FEN_{cncip}$), we analyze the failing queries that occur in the vicinity of a successful DNS query. Our hypothesis is as follows. *For a bot issuing a burst of DNS queries to determine the C&C server address, the entropy of failed DNS queries present in the burst, is of the same order as the entropy of the successful queries*. We again use the normalized edit distance for determining the entropy of failed domain names present in a time bin containing successful query resolution. It is symmetric about the time instant where $cncip$ was observed. It is noteworthy that all failed DNS queries present in the time bin, may not be related to the successful DNS query. Such queries deviate the output. The noise is especially amplified at DNS *aggregators* which query on behalf of several individual local clients. Thus, choosing an appropriate time window length is critical for accurate analysis. During evaluation, we show how changing window size affects the performance.

To compute the failed domain entropy (FEN) for a candidate C&C IP address, we use the following equation:

$$FEN_{cncip} = \frac{\sum(FEN_{client})}{\text{Number of clients}} \tag{2}$$

where $FEN_{client}$ is the FEN value computed by examining *client's* time series of query generation, with respect to $cncip$. To elaborate, the failed query entropy for a client is computed between pairs of strings (failed domain names) present within every time window in which $cncip$ occurs. Subsequently, all such FEN values are averaged thus giving $FEN_{cncip}$. The computation of this entropy requires at least two failed domain names within the window of consideration. A higher number of failures increase the confidence in the computed $FEN$ value for that window implying that botnets are detected more accurately. Alternately, individual failed queries can be directly compared with the candidate successful domain names to compute the edit distance relevant to each failed domain name. We have evaluated both approaches and obtained similar results. Owing to space constraints, we only report on one of the approaches.

To further filter the candidate set of anomalous IP addresses, we consider only those addresses with $FEN_{cncip}$ greater than a threshold. We choose a conservative threshold to avoid ignoring genuine anomalies. To apply our hypothesis of the proximity of $SEN_{cncip}$ and $FEN_{cncip}$, we use the following inequality to further eliminate false positives:

$$(SEN_{cncip} - \delta) \leq FEN_{cncip} \leq (SEN_{cncip} + \delta) \tag{3}$$

where $\delta$ represents a small bound or the *proximity* within which $FEN_{cncip}$ and $SEN_{cncip}$ are expected to lie. To choose an appropriate $\delta$, we compute the standard deviation $\sigma$ of entropy for domains belonging to the known botnet IPs present in our dataset. Thereby, we choose $\delta$ as $3\sigma$.

From the above description, the temporal correlation and entropy related parameters are analyzed and IPs satisfying the outlined malicious criteria, help in identifying bots within the network as well. In an autonomous system, where the DNS queries are observed from local clients and DNS aggregators, our technique may be applied recursively at the aggregator, yielding the bots which use the aggregator as their DNS recursive resolver.

### 3.2 Correlating failures for improved latency

Here, we present an alternate strategy for *speeding* up the detection technique which relies only on successful queries. The work in [11] emphasizes upon applying statistical techniques such as K-L divergence, Jaccard Index, and Edit distance, to the set of successful domains for a $cncip$ ($SQ_{cncip}$). Through evaluation, it is shown that a large set improves the accuracy of anomaly detection. However, accumulating a larger set requires a considerable amount of time. Therefore, we propose supplementing the set $SQ_{cncip}$ with failed queries that occur within the vicinity of successful DNS queries. The resulting set is accumulated faster, decreasing the latency of analysis by an order of magnitude.

The detection technique proposed in this work provides the basis for associating the queries with successes viz. temporally. In addition to temporal characteristics, to improve the quality of failed domain name set, we propose considering only those failing domains within the time window, whose entropy characteristics are similar to the successful domain set. Such similarity parameters for entropy can help identify a DGA. Here, we explore supplementing $SQ_{cncip}$ with these two main features. Additional features as described for detection can also be used.

To realize the faster accumulation of relevant domains, we compute the entropy (normalized edit distance) between a failed domain name and each of the successful DNS domain names discovered under analysis. A domain yielding an entropy value close to the average $SEN$ (or the entropy of successful domains) is considered relevant. The measure of closeness is defined using eqn. 3 as described above. For this particular experiment, we choose $\delta = \sigma$. We note that such marginally expensive computation for identifying relevant domains may improve latency *and* accuracy. We evaluate this correlation strategy for Conficker's C&C server addresses in section 4. Note that the analysis of discarded failed domain names

**Table 1.** Trace description.

| | ISP trace | Campus trace |
|---|---|---|
| **Trace collection period** | Nov 03-04, 2009 | Aug 22 - Sep 22, 2010 |
| **Total number of DNS sessions** | 1.61 M | 112.7 M |
| **White-listed sessions** | 770.23 K | 54.4 M |
| **Total number of failed DNS packets after white-listing** | 57.72 K | 1.28 M |
| **IP addresses analyzed for maliciousness** | 9948 | 74.7 K (per segment avg.) |
| **Number of clients (or aggregators)** | 8472 | 1735 (per segment avg.) |

identifies irrelevant queries belonging to services like *qq.com* and *ask.com* while few excluded domains belong to Conficker, strengthening our confidence in the new set.

## 4 Results

We validate our technique using the DNS datasets described below. For our analysis, we consider only DNS type A records. Several DNS blacklist based services utilize the A record to verify whether an IP address, domain name, or an executable (a feature used by McAfee) is present in the blacklists. To exclude these queries from analysis, we white-list a total of 31 trusted second-level domain names including several blacklist services, Content Distribution Network services (such as *akamai.net, cloudfront.net*) and popular domains (such as *google.com, facebook.com*). The white-list helps us focus on other potentially malicious domains, in addition to refining the failed domains set used for analysis. Additionally, we avoid processing answers with RFC 1918 (private) addresses [7].

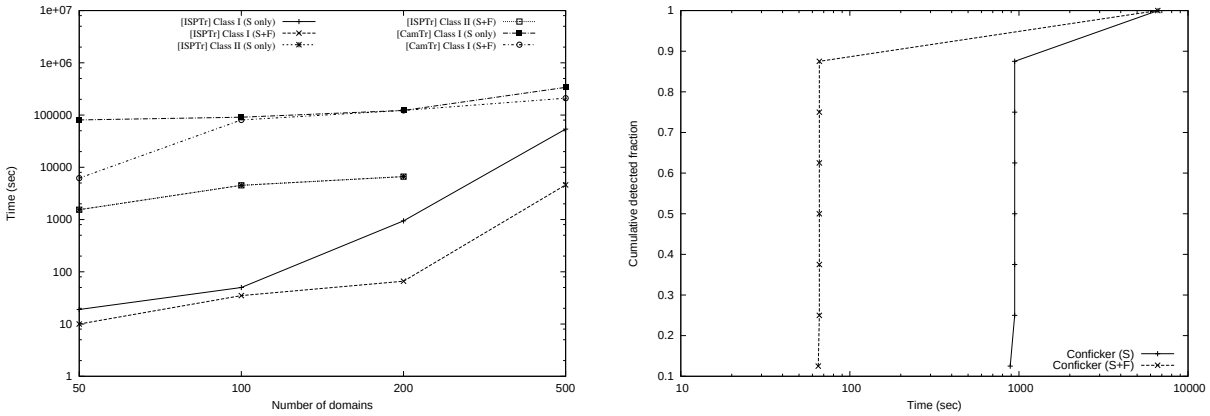
### 4.1 Data sets

Table 1 details the traces used for analysis. The 20-hour long ISP trace contains known malicious IPs belonging to the Conficker botnet. Using a blacklist, we obtain a set of 100 odd IPs labelled malicious, which we further verify manually by checking against exhaustive databases such as *robtex.com* and *mywot.com*. We believe that these two sources provide us with the most recent information concerning the queried domains or IP addresses. The 19 IPs obtained post verification with the above sources, contain two IP addresses hosting adult websites. We disregard these as non-fluxing behavior. One C&C address apparently belongs to the domain-fluxing Kraken/Bobax botnet (based on the domain names we see). However, the Kraken C&C address has a degree of only two. The 16 remaining IPs belong to Conficker, some of which are sinkhole servers [1]. Nonetheless, we consider them as anomalous as they help keep the botnet alive. As a result, we consider the remaining 9931 IPs as legitimate. Note that all IPs considered for ground truth evaluation, have a degree of at least two.

We also use a DNS trace captured at a primary recursive resolver of a university network. We divide the month-long trace into approximately week-long segments and present our results on randomly chosen segments. Each segment contains an average of 295K IP addresses returned as DNS responses (after white-listing). As table 1 shows, approximately 75K IP addresses have degree $D_{cncip} > 2$. For the campus trace, we use the C&C server information from the ISP trace to obtain 29 IP addresses (out of 75K), labelled as malicious. Since the ground truth information for the ISP trace is relatively old, we again verify this set manually. As a result, we are left with *four* Conficker C&C addresses which are common with those present in the ISP trace.

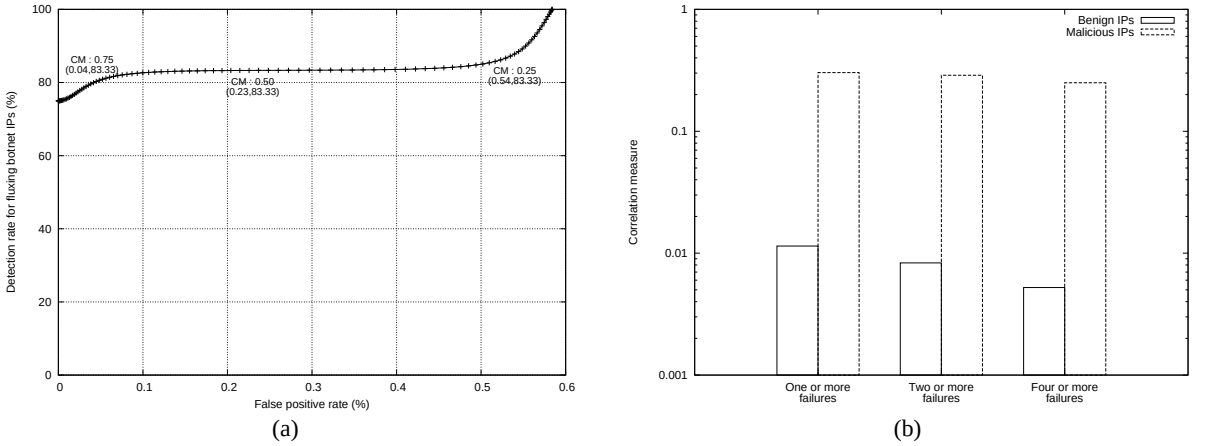
### 4.2 Latency comparison

The latency of detection is expressed in terms of the number of successful domain names required to analyze and detect a rogue server accurately. Figure 2(a) shows the gain obtained in terms of time taken to collect a set of botnet domain names. The figure shows two classes of botnet IPs that we observe. Class I represents those C&C server addresses where domain fluxing yielded both successful and failed queries. However, for the C&C server in Class II, the bots issued none or very few failed DNS queries. In our ISP trace with more than 50 domain names mapping to it, we find eight C&C server addresses belonging to




(a) (b)

**Fig. 2.** Latency comparison (a) for different number of domain names. (b) for 200 domain names.




(a) (b)

**Fig. 3.** (a) ROC curve for changing correlation thresholds. (b) Correlation comparison for changing flux behavior.

Class I and two belonging to Class II. Also, we observe that all four C&C addresses in the analyzed campus trace segment belong to Class I.

Figure 2(a) shows the improved latency for the average time taken by Class I or II addresses. From the figure, we observe that when failed domain names are correlated with successful DNS domains, the time taken to collect 50, 100, 200 or 500 domain names is considerably reduced. Especially, for 500 domain names, we see a gain of an order of magnitude when the time of collection reduces from approximately 54000 secs to only 4600 secs. With Class II, however, we do not observe any gain since no failures help in supplementing the set of successful botnet domains. Thus, we infer that in context of applying statistical techniques for anomaly detection, the proposed correlation mechanism can significantly reduce the time to collect input required for analysis of Class I C&C addresses. Note that we do not obtain 500 successful domains for the Class II address. The analysis with the campus trace follows analogous behavior. However, we plot the latency observed when correlating failures with successes, using the criterion highlighted in section 3.2. We also note a higher initial latency for domain name collection, owing to slower traffic seen for a campus (Tier-4) as compared to a Tier-1 network trace. Although, the time of collection is reduced considerably even with the campus trace. For instance, we observe 100 domains are collected 10K seconds faster than when using only successful DNS traffic.

While figure 2(a) shows average time taken for Class I and II anomalous entities, figure 2(b) shows the pace at which those anomalies are detected, for the specific case of 200 domain names. From the figure,

we see that using the both successful and failed DNS domains, we can detect more than 80% of the total IP addresses, an order of magnitude faster than when using only the successful ones. We note that the cumulative detection reaches 100% because of the presence of Class II address.

From the figures, we conclude that considering failed domain names assists in speeding up detection of domain-fluxing botnet. While speeding up the detection through methods presented in [11], the worst case detection latency is same as the original latency where only domains from $SQ_{cncip}$ are used. We also transform botnet detection to a real-time detection approach through the speeding mechanism presented above, as well as through the detection strategy.

### 4.3 Effect of the correlation parameter

We evaluate the significance of using the correlation as a feature for anomaly detection. Figure 3(a) represents the Receiver Operating Curve (ROC) curve for changing thresholds for correlation between DNS successes and failures. The ROC curve shows a decrease in false positive rate with a decreasing detection rate, when increasing $Corr_{cncip}$ thresholds. A higher threshold requirement would imply that failures coincide with the successful queries more frequently. We would expect benign IP addresses to have a low correlation value. Hence, increasing the threshold results in decreasing false positives. For this particular experiment, we note detecting a maximum of 12 (out of 17 C&C IPs) as the remaining IPs do not have enough domain names for analysis (at least eight domains). We also note a maximum false positive rate of only 0.6% which primarily comprises of legitimate IPs with relatively lower entropy than seen for fluxing botnets and few failures within the window of analysis (that is, a correlation value just above the threshold). In contrast, fluxing botnets usually have a high number of failed domain names within the corresponding bin. Note that the false positives include ISP's intra-AS DNS resolution queries where the hosts have been assigned random-appearing domain names.

### 4.4 Correlation vs Number of DNS failures

Through this experiment, we aim to study the behavior seen for malicious IPs, in terms of the number of failed domain names generated. Figure 3(b) shows the correlation observed for malicious and benign IPs. We compute the correlation for three cases where we expect at least one, two or four failures to occur within the same window, as the $cncip$. The figure shows a decrease in correlation, for both benign and legitimate IPs, though the correlation reduces only slightly for the malicious set. Such a study implies that the correlation criteria may be adapted towards highly fluxing botnets while keeping fewer false positives.

### 4.5 Variation in entropy

We evaluate the impact of information entropy expressed by the domains which map to a candidate $cncip$ (that is, $SEN_{cncip}$). The effect of changing the entropy thresholds for considering a $cncip$ is shown through an ROC curve as in Figure 4. The figure shows an increase in the detection rate and the false positive rate as the threshold for entropy is decreased. This is in line with the observation that for a low threshold, several sets of domain names, and in particular those for CDNs satisfy the filters used for detection. Analogous to the performance with varying correlation threshold, the maximum false positive rate observed is 0.52%. Analysis of false positives reveals DNSBL services (*redcondor*), popular websites (*sina.com.cn* which offer multiple services), DNS and HTTP servers providing service to multiple entities, blogging services (*blogspot*) and CDN addresses. For instance, we observe *redcondor* IPs labelled malicious when the entropy thresholds are 0.35 or lower, indicating that even though correlation may be high for this DNSBL service, the entropy helps distinguish it from actual anomalies, when using higher thresholds. In section 5, we discuss how botnets may attempt to fool our detection approach into generating domains with low entropy. Note that we determine the detection rate over the 12 detectable botnet IPs, as discussed previously.

### 4.6 Size of time bin

The impact of varying the size of time bins is shown in Table 2. In all experiments, we observe that false positives increase with larger time bins. The wider bins allow a higher possibility of inclusion of failures within the corresponding $cncip$ bin, resulting in an increased $Corr_{cncip}$ value. Thus, more candidate IP

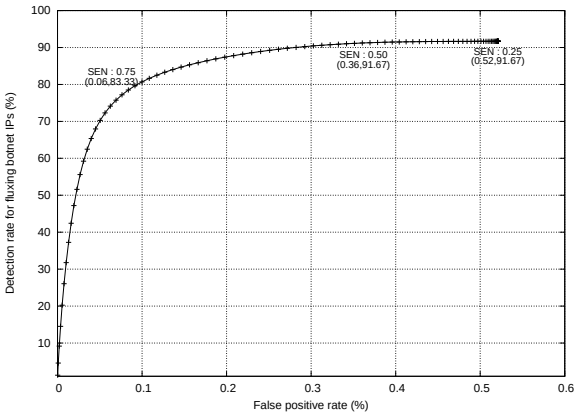



**Fig. 4.** Performance with changing entropy thresholds

**Table 2.** Impact of changing time bin size.

| Window size (sec) | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|
| | **ISP trace** | | | | | | |
| **FPR (%)** | 0.022 | 0.043 | 0.043 | 0.097 | 0.173 | 0.259 | 0.302 |
| **TPR (%)** | 75.0 | 75.0 | 75.0 | 75.0 | 75.0 | 83.33 | 83.33 |
| | **Campus trace** | | | | | | |
| **FPR (%)** | 0.021 | 0.039 | 0.084 | 0.120 | 0.209 | 0.434 | 0.752 |
| **TPR (%)** | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |

addresses meet the criteria set by filter $F_2$ (in figure 1) and the ones with high entropy may be incorrectly labelled malicious. From the table, we also observe that smaller window sizes result in fewer IP addresses being detected within the ISP trace. An inference of this observation is that bots spread their DNS queries making temporal correlation between failures and successes difficult. The false positive rate, however, is less than 0.3% for ISP trace (and 0.75% for campus trace) for all experiments making the choice of larger windows possible. However, the marginal or no increase in detection rate and false positive rate, for the change in window size from 4 secs to 128 secs implies that a window as small as 4 secs may be sufficient for detecting a domain-fluxing botnet like Conficker. Note that the false positives for the campus trace are higher when compared to the ISP trace, as we frequently observe DNSBL based NXDOMAIN failures within the campus dataset. Such DNSBL failures affect the correlation/entropy parameters resulting in more benign IPs classified as malicious.

*Dyndns* is a service for generating customizable/random temporary domain names. In the ISP trace, we observe several random sub-domains for *dyndns* which exhibit high entropy characteristics, with several failures belonging to this service. Our detection mechanism designates these IP addresses as benign owing to presence of temporally distant failures, reinforcing our hypothesis of utilizing temporal correlation of failed domains for anomaly detection through the use of small time bins. We also limit our study to a maximum bin size of 256 seconds as fast-fluxing botnets have low DNS Time-To-Live (TTL) values.

## 5 Discussion

In this work, we detect Conficker C&C addresses which exhibit high entropy owing to randomized distribution of alphanumeric characters composing the domain names. However, to evade our detection mechanism, botnet owners may alter the way domain names are composed. For instance, our separate study has observed combination of dictionary words being used as an alternate way of domain-fluxing. Some example domain names that we observe for a botnet are *haireconomy.ru, greedycake.ru* and *empirekey.ru* [2]. The information entropy computed for a set of such domain names indicates that owing to high edit distance values, such domain names can still be distinguished through entropy analysis.

To validate the robustness of our approach, we artificially inject domain queries as observed above, with some of them failing and some domain names successfully mapping to a reserved address. We randomly choose clients to insert such DNS queries. Based on our study, we still detect the simulated anomalies with similar experiment parameters as used previously for evaluation. Also, in future if botnet owners formulate a DGA where the observed entropy is lower than that observed for fluxing botnet detected in this work, our detection mechanism can detect them with low false positives, as hinted by figure 4.

A direct weakness of our detection strategy is reliance on failed domain names. Our experiments are based on analyzing the first few successful domain names and thus correlating failures that are present in their vicinity. In the event that no failures are present, or failures that occur right after the window of

analysis, our detection strategy may fail in which case switching to the algorithm for correlating failures to *supplement* the set $SQ_{cncip}$ would help. Thus, a combination of both the strategies presented in this work may be useful for fast anomaly detection. It is possible to generate DNS queries slowly such that the failed queries are outside the time window considered in our scheme. Such an approach, however, will slow down the bot in identifying its C&C server and hence constraining the botnet writer again.

Our technique for detection can be mapped back to detect individual bots that issued the queries for a malicious $cncip$. For instance, with our campus trace analysis, we observe 12 hosts within our AS querying for three of the four C&C addresses, and 10 hosts (subset of the 12 above) querying for the remaining C&C address. While the Tier-1 ISP trace may not have individual clients (we would mostly observe aggregators), the mechanism applied for a smaller-sized network or when applied recursively, may result in more accurate detection of anomalies and bots, owing to a better DNS failure signal.

## 6 Conclusion

In this paper, we proposed methodologies for utilizing failed domain names in the quest for rapid detection of a fluxing botnet's C&C server, the bots within the local network, and the related domain names, and thus revealing the botnet infrastructure. Utilizing only DNS traffic, we reduce the resource requirement for botnet detection. We also considerably reduce the latency of detection when compared to previous techniques. For faster detection, we utilize not only the entropy of the domain names successfully mapping to an IP address, but also that of the correlated failed DNS queries occurring within the vicinity of the succeeding DNS query. With our technique, we achieve a false positive rate as low as 0.02% with a high detection rate. As a future work, we plan to utilize SERVER FAILURE based DNS failures, or failures related to the name servers, as a means for detecting botnets which exhibit double fast flux.

## References


1. Conficker Working Group. `http://www.confickerworkinggroup.org/wiki/pmwiki.php/ANY/FAQ#toc5`.
2. New Technique Spots Sneaky Botnets. `http://mobile.darkreading.com/9292/show/4711c9403b772e7281ae08cee69758cc\&t=461a4a89abc0a0c761234d11086f5003`.
3. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. *Proc. of the 16th USENIX Security Symposium (Security'07)*, Aug. 2007.
4. N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang. Identifying Suspicious Activities Through DNS Failure Graph Analysis. *IEEE Conference on Network Protocols*, 2010.
5. C. D. Manning, P. Raghavan, and H. Schutze. An Information to Information Retrieval. *Cambridge University Press*, 2009.
6. P. Porras, H. Saidi, and V. Yegneswaran. Conficker C Analysis. Technical report. `http://mtc.sri.com/Conficker/addendumC/`.
7. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear. Address Allocation for Private Internets, 1996. `http://www.ietf.org/rfc/rfc1918.txt`.
8. 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 Communications Security (CCS)*, Nov 2009.
9. R. Villamarín-Salomón and J. C. Brustoloni. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic. *Consumer Communications and Networking Conference*, 2008.
10. R. Villamarín-Salomón and J. C. Brustoloni. Bayesian Bot Detection Based on DNS Traffic Similarity. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 2035–2041, New York, NY, USA, 2009. ACM.
11. S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. *Internet Measurement Conference*, 2010.
12. S. Yadav and A. N. Reddy. MiND : Misdirected dNs packet Detector. *IASTED Computer and Information Security*, 2010.
13. Z. Zhu, V. Yegneswaran, and Y. Chen. Using Failure Information Analysis to Detect Enterprise Zombies. *Security and Privacy in Communication Networks*, 2009.