Early Detection of Malicious *Flux* Networks via Large-Scale Passive DNS Traffic Analysis

Roberto Perdisci, Igino Corona, and Giorgio Giacinto

Abstract—In this paper we present FluxBuster, a novel *passive* DNS traffic analysis system for detecting and tracking malicious *flux* networks. FluxBuster applies *large-scale* monitoring of DNS traffic traces generated by recursive DNS (RDNS) servers located in hundreds of different networks scattered across several different geographical locations. Unlike most previous work, our detection approach is not limited to the analysis of suspicious domain names extracted from spam emails or precompiled domain blacklists. Instead, FluxBuster is able to detect malicious flux service networks *in-the-wild*, i.e., as they are "accessed" by users who fall victim of malicious content, independently of how this malicious content was advertised. We performed a long-term evaluation of our system spanning a period of about five months. The experimental results show that FluxBuster is able to accurately detect malicious flux networks with a low false positive rate. Furthermore, we show that in many cases FluxBuster is able to detect malicious flux domains several days or even weeks before they appear in public domain blacklists.

Index Terms—Flux Networks, DNS, Passive Traffic Analysis, Clustering, Classification, Internet Security

1 Introduction

Internet miscreants and cyber-criminals are always looking for new ways to cover the traces of their malicious activities while preserving their illicit revenues. To this end, malicious *flux* networks have recently started to thrive [1]. Malicious flux networks can be viewed as *illegitimate* content-delivery networks (CDNs). Legitimate CDNs have been used for quite some time to provide a high degree of availability, scalability, and performance to legitimate high-volume Internet services. A CDN usually consists of a relatively large number of nodes scattered in multiple locations around the world. Whenever a user requests a service provided through a CDN, the CDN's node closest (non necessarily in a geographic sense) to the user is usually chosen to provide the requested content with high performance. CDNs can be implemented using the DNS, whereby domain names related to the content network resolve to multiple IP addresses related to the best node for each user's request. Unlike legitimate CDNs, whose nodes are professionally administered machines, the nodes of a malicious flux network, a.k.a. *flux agents*, are typically malware-compromised machines. The flux-agents are usually part of a *botnet* – a network of malware-compromised machines that can be commanded to perpetrate malicious actions in a coordinate way – and can be remotely controlled by an adversary, who is often referred to as the *botmaster*. Malicious flux networks are commonly used to host phishing websites, illegal adult content, or serve as malware propagation vectors, among other things.

The main technical difference between a malicious *flux* network and a legitimate CDN is that, while the nodes of a legitimate CDN are highly reliable and tightly controlled by the CDN administrators, botmasters do not usually have complete control over the flux agents. Many of the compromised machines that form a malicious *flux* network may be turned on and off by their owners at any time, making the uptime of each *flux* agent hard to predict. Also, differently from CDNs, it may be hard for the botmaster to tightly monitor the load on each node, and to redistribute the received content requests accordingly. In order to cope with these problems and maintain high content availability, botmasters usually set up their malicious networks using *fast-flux* domain names. The term *fast-flux* is used to point out that the set of resolved IP addresses (the flux agents) associated to the malicious domain names change frequently, often at each DNS query [2]. Furthermore, since it is usually hard for the botmaster to control exactly where the malware propagates and which machines are infected by her *bot* software, the flux agents are often scattered across many different networks [2].

A number of approaches for detecting fast-flux domain names have been recently proposed [3], [4], [5], [6]. These works (which will be discussed in more details in Section 2) differ from each other in the number of features used to characterized fast flux domains and the details of the classification algorithms, but they all rely on an *active-probing* approach, whereby domains extracted from spam emails or malware domain blacklists are repeatedly queried to collect their sets of resolved

- *Roberto Perdisci is with the Department of Computer Science, University of Georgia, Athens, GA 30602, USA. E-mail: perdisci@cs.uga.edu*
- *Igino Corona is with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, ITALY E-mail: igino.corona@diee.unica.it*
- *Giorgio Giacinto is with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, ITALY E-mail: giacinto@diee.unica.it*

IPs. This information is then used to classify each domain name as being either *fast-flux* or *non-fast-flux*.

In this paper we present FluxBuster, a novel detection system that uses a *purely passive* approach for detecting and tracking malicious flux networks. Our detection system is based on *large-scale* passive analysis of DNS traffic generated by hundreds of local recursive DNS (RDNS) servers located in different networks and scattered across several different geographical locations (see Figure 1). FluxBuster is based on our previous work [7], but differs from it in several aspects:

- Our previous study [7] was based on DNS traffic observation from "below" a number of recursive DNS servers, namely DNS traffic flowing from single users' machines to their local RDNS servers. Unfortunately, obtaining access to such a type of traffic is problematic, mainly due to privacy concerns. In [7] we were able to access this "below-recursive" traffic thanks to the collaboration with a security company and under a non-disclosure agreement. In this work we study the detection of flux networks by passively monitoring DNS traffic collected from "above" local RDNS servers, as shown in Figure 1. This monitoring approach is *privacy-preserving*, because the IP addresses of the individual clients are masked by their local RDNS server. Furthermore, the RDNS server acts as a *traffic mixer*, and the DNS cache has the effect of suppressing queries to domain names whose TTL has not yet expired. This makes the proposed system easier to be adopted, as the data on which it is based can be safely shared by network operators, for example through the Internet Systems Consortium's Security Information Exchange (ISC/SIE) frameworks [8].
- Because we use a different type of data source than the one we used in [7], we had to significantly change the way FluxBuster analyzes the DNS traces and extracts the statistical features needed to classify flux networks.
- Finally, this paper presents an evaluation of our detection system at a much larger scale and over a much larger time period (about five months).

FluxBuster's detection approach can be summarized as follows. Because the amount of RDNS traffic in large networks is often overwhelming, we first apply a number of pre-filtering rules that allow us to quickly discard DNS messages that are clearly related to non-flux domains (see Figure 2). At the same time, these pre-filtering rules will not discard information about domain names that are actually related to *live* malicious flux networks. In practice, the output of the pre-filtering stage is a set of *candidate* flux domains. This approach drastically reduces the volume of DNS traffic to which we apply a more expensive, fine-grained analysis. It is worth noting that this pre-filtering stage is very conservative, and will not exclude legitimate Internet services that share some characteristics with flux networks, such as CDNs, NTP pools, etc. To distinguish between *malicious flux* and *non-flux* networks, given the set of candidate malicious flux domains we perform a more fine-grained analysis by first grouping together domain names that are related to each other. For example, we group together domain names that either point to the same Internet service, are related to the same CDN, or are part of the same malicious flux network. In practice, each of these domain clusters represents a *candidate* flux network. We then use a statistical classifier to detect whether a domain cluster is actually related to a malicious flux network or to a non-flux network (Section 3.1). This is in contrast with most previous works, in which single domain names are considered independently from each other, and classified as either fast-flux or non-fast-flux [3], [4], [6], thus ignoring the fact that many flux networks involve more than one domain name.

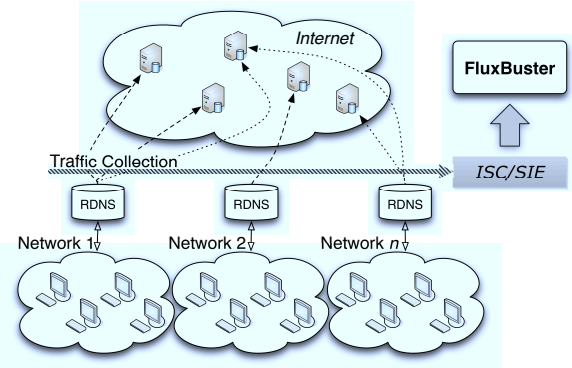

Fig. 1. Traffic collection "above" local DNS servers.

We have implemented a proof-of-concept version of FluxBuster and evaluated it on DNS traffic shared by hundreds of different network operators through the ISC/SIE framework [8]. Our long-term evaluation spans a period of about five months, and shows that our system is able to accurately detect malicious flux networks with a low false positive rate. Furthermore, we show that in many cases our system is able to detect malicious flux domains several days or even weeks before they appear in public domain blacklists. In addition, we have made our results available to the security community, and have already received positive feedback from a number of security researchers and professionals.

2 Related Work

A number of approaches for detecting fast-flux domain names have been recently studied in [3], [4], [5], [6], [9], [10]. These works differ from each other in the number of features used to characterized fast flux domains, and the details of the classification algorithms. The main limitation of these works lies in the use of spam email as the primary information source, thus detecting fast-

flux domains advertised through email spam[1]. These approaches identify potential fast-flux domain names in the URLs found in the body of spam emails (typically captured by spam traps and filters) [3], [4], [5], [6]. Then, an *active probing* strategy is applied, by repeatedly issuing DNS queries to collect information about the set of resolved IP addresses, and by subsequently classifying each domain name as being either *fast-flux* or *non-fast-flux*. The work in [9] is in part different from other previous works, because it is not limited to domains found in spam emails. In particular, they propose to analyze NetFlow information collected at border routers to identify *redirection botnets*, which are a specific kind of botnets used to set up *redirection* flux service networks. However, the information they extract from network flows is not able to detect flux agents that are being used as *transparent proxies*, instead of redirection points. In addition, to perform the classification of suspicious domains collected from spam emails, and the correlation with information regarding network flows, this work heavily relies on DNS active probing in a way very similar to [3], [4].

Our detection approach, based on passive monitoring, has a clear advantage over detection techniques proposed in previous works. Passively monitoring *live* users' DNS traffic allows capturing queries to flux domain names that are advertised through a variety of means, including, for example, *blog spam*, *social websites spam*, *search engine spam*, and *instant messaging spam*, in addition to email spam and precompiled domain blacklists such as the ones used in [3], [4], [5], [6]. Furthermore, unlike the active probing approach used in previous work [3], [4], [5], [6], we passively monitor live users' traffic without interactions with the flux networks. Active probing of fast-flux domain names [3], [4], [5], [6] may be detected by the attacker, who often controls the authoritative name servers responsible for responding to DNS queries about her fast-flux domain names. If the attacker detects that an active probing system is trying to track her malicious flux service network, she may stop responding to queries coming from the probing system to prevent unveiling further information. On the other hand, our detection system is able to detect flux services in a *stealthy* way.

Recently Hsu et al. proposed a real-time system for detecting flux domains based on anomalous delays in HTTP/HTTPS requests from a given client [10]. The assumption is that the flux agents are often used as either web servers or web proxies to provide malicious content (e.g., phishing web pages). Because the flux agents are typically malware-compromised home machines, rather than performant web server, they often provide the malicious web content with large latencies. Hsu et al. leverage on these observations for detection purposes.

1. Some works also consider domain blacklists and malware samples as information source, but the number of collected domains is quite small if compared to the number of domain names that are extracted from spam emails.

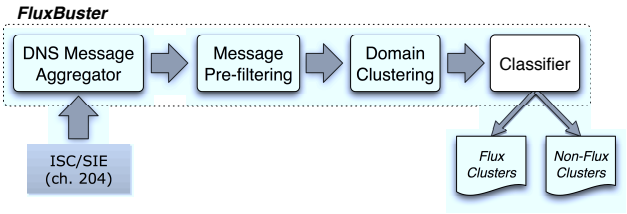

Fig. 2. High-level system overview.

Our work is significantly different from [10], which focuses mainly on HTTP traffic generated by single clients and may not scale well in very large networks. Moreover, the work in [10] employes a mix of passive and active approaches to detect flux domains. On the other hand, our work focuses on large-scale, privacy-preserving passive analysis of DNS traffic, and does not need access to the HTTP traffic generated by single clients, which may be very difficult to obtain due to privacy concerns.

A number of DNS reputation systems mainly targeted at detecting *generic* malicious domains [11], [12], or malware-specific domains [13] have been recently proposed. These works have in common the use of large-scale DNS monitoring to detect malicious domains. Our work is different because FluxBuster aims to detect and track flux networks, rather than focus on single low-reputation domain names, thus providing a more targeted and accurate output for this specific type of maliciousness.

3 FLUXBUSTER

In this section we describe how FluxBuster works. We start by giving a high-level overview of our system, before describing its components in more detail.

3.1 System Overview

Figure 2 shows a high-level overview of FluxBuster. FluxBuster receives in input a stream of DNS messages collected through the ISC/SIE [8] infrastructure, as shown in Figure 1. The details of the data collection process are discussed in more details in Section 3.4. For the sake of the following discussion, it is important to notice that each collected DNS message provides information regarding the mapping of a domain name d to a set of resolved IP addresses R, as observed from the ISC/SIE DNS sensors distributed across the Internet.

The *DNS Message Aggregator* module (Section 3.2) aggregates all DNS messages regarding a domain d into a higher-level DNS message $Q(d)$ that contains all information about d observed during a given interval of time Δ. Each $Q(d)$ includes information such as how many DNS queries to d were observed during Δ, the aggregated set of all resolved IP addresses pointed by d, their average *time to live* (TTL), etc. The length of Δ should be at least a few hours, so that we can collect

enough information (e.g., several resolved IPs) for each flux domain name (see Section 3.2).

The *Message Pre-filtering* module (Section 3.4) analyzes each aggregated DNS message $Q(d)$ and filters out those messages related to domains d that are very unlikely to be flux domains. In practice this module performs *data volume reduction*, and aims to reduce the computational cost of the more fine-grained analysis performed by the following modules. To this end, we implement the *Message Pre-filtering* module using a set of very conservative filtering rules. The consequence is that at this stage FluxBuster "accepts" all the flux domains as well as several non-flux domains, such as domains related to CDNs, as *candidate flux domains* for further analysis.

The set of domains $D = \{d_1, d_2, ..., d_n\}$ and the related set of aggregated messages $Q(d_1), Q(d_2), ..., Q(d_n)$ that pass the pre-filtering step are then sent to the *Domain Clustering* module (Section 3.5). This module partitions the set of domains D into a set of domain clusters $C = \{C_1, C_2, ..., C_m\}$, with $D = \bigcup_i C_i$. Each of these clusters contains domains that are related to each other, because they share a non-negligible percentage of resolved IPs. In practice, these clusters of domains and the related sets of IP addresses represent *candidate flux networks*.

Because of the conservative filtering approach taken by the *Message Pre-filtering* module, not all these candidate flux networks are actually *flux*. Therefore, in order to distinguish between flux and non-flux networks, FluxBuster employs a *Classifier* module (Section 3.7) based on a supervised statistical algorithm that has been previously trained on examples of both flux and non-flux networks. The *Classifier* module processes the clusters produced by the *Domain Clustering* module that include a number of IP addresses larger than a predefined threshold κ (e.g., $\kappa = 30$), and labels them as either *flux* or *non-flux*. We use the threshold κ first of all because clusters with very few IP addresses are extremely unlikely to represent active flux networks, and because we need a sufficiently large and representative sample set of IP addresses for each candidate flux network in order to compute some of the statistical features described in Section 3.6. Therefore, clusters that contain a number of IP addresses lower than κ cannot be reliably labeled as *flux*, and are simply discarded.

3.2 DNS Message Aggregator

FluxBuster receives in input a stream of DNS messages as provided by the ISC/SIE [8] framework. ISC/SIE collects raw DNS query/response messages from a large number of RDNS sensors (see Figure 1), and re-broadcasts these DNS messages in a *de-duplicated* fashion[2]. For example, assume that there are three RDNS sensors S_1, S_2, and S_3 that have reported a DNS query/response message regarding a domain name d to ISC/SIE. Suppose that S_1 reported the mapping of domain d to three IP addresses, $\{IP_1, IP_2, IP_3\}$, S_2 reported the mapping of d to two addresses $\{IP_1, IP_4\}$, and S_3 reported the mapping of d to one address $\{IP_5\}$. These raw messages will be combined within the ISC/SIE framework into a de-duplicated message stating that d maps to $\{IP_1, IP_2, IP_3, IP_4, IP_5\}$. In other words, the de-duplication process aggregates raw DNS query/response messages received within a certain time window T into a single message that summarizes the domain-to-IP mappings observed during T from multiple sensors. Because of the implementation of the ISC/SIE de-duplication system, the value of T varies with time, and it is in the order of a few hours (e.g., two to four hours).

FluxBuster performs a further aggregation of the messages that are received from ISC/SIE within a period Δ equal to twelve hours. The aggregated message can be represented as a tuple $Q(d) = \{d, t_1, t_2, R, \tau, c\}$, where d is a given domain name, t_1 and t_2 represent the timestamp of the first and last DNS query/response messages regarding d that were aggregated into $Q(d)$, respectively, R is the set of all IP addresses resolved during the period $(t_2 - t_1)$, τ is the average time-to-live (TTL) of the DNS responses, and c is the number of raw DNS query/response messages aggregated by $Q(d)$.

3.3 Characteristics of Flux Domain Names

Before presenting the *Message Pre-Filtering* module, we discuss the typical characteristics of flux domains, which we used to derive our pre-filtering rules.

Fast-flux domains are characterized by the following main characteristics: (a) short time-to-live (TTL); (b) high frequency of change of the set of resolved IPs (i.e., the flux agents) returned at each query; (c) the overall set of resolved IPs obtained by querying the same domain name over time is often very large; (d) the resolved IPs are scattered across many different networks [2]. Some legitimate services, such as legitimate CDNs, NTP server pools, IRC server pools, etc., are served through sets of domain names that share some similarities with fast-flux domains. For example, domains related to legitimate CDNs often have a very low TTL and resolve to multiple IP addresses located in different networks. Also, domains related to NTP server pools use a very high number of IP addresses which change periodically using a round-robin-like algorithm. Although the value of each individual characteristic may not allow us to precisely identify malicious flux domains and distinguish them from legitimate domains, combining multiple measures based on the above characteristics enables accurate detection of flux domains and networks (see Section 3.6).

3.4 Message Pre-Filtering

The *Message Pre-Filtering* module performs data volume reduction by discarding domain names that are very unlikely to be part of a flux network.

2. FluxBuster currently receives de-duplicated DNS messages from SIE channel 204.

Given an aggregated DNS message $Q(d)$ related to a domain d, $Q(d)$ is "accepted" by the filter, i.e. it is considered for further analysis, if *all* the following rules are matched:

(1) $\tau \leq \theta_{ttl}^{'}$
(2) $R \geq \theta_R$ OR $\tau \leq \theta_{ttl}^{''}$
(3) $div(R) \geq \theta_{div}$

where $\theta_{ttl}^{'}$, $\theta_{ttl}^{''}$, θ_R, and θ_{div} are suitably chosen thresholds (with $\theta_{ttl}^{''} < \theta_{ttl}^{'}$). The function $div(R)$ computes the "diversity" of the resolved IP set R, and it is computed as $div(R) = \frac{|P|}{|R|}$, where P is the set of all the /16 IP prefixes in R. P is computed by considering each IP in R, and extracting its /16 prefix (e.g., the /16 prefix of 128.192.76.177 is 128.192). Thus it follows that $|P|$ is the number of distinct IP prefixes in R, and a large value of $div(R)$ is associated to a set of resolved IPs that are scattered across many different /16 networks (we use /16 prefixes because they tend to approximate well the boundary among networks belonging to different organizations, as discussed in Section 3.6).

The choice of these simple filtering rules follows directly from the characteristics of flux domains outlined in Section 3.3. To make sure that the *Message Pre-Filtering* module does not discard flux domains, we conservatively set the filtering thresholds as follows: $\theta_{ttl}^{'} = 3$ hours, $\theta_R = 3$, $\theta_{ttl}^{''} = 30$ seconds, and $\theta_{div} = \frac{1}{3}$. Therefore, only domains with a very large TTL, very low number of resolved IPs, and a low value of diversity of the IP set will be discarded (i.e., they will be regarded as non-flux domains). It is also worth noting that the threshold $\theta_{ttl}^{''}$ is used to avoid discarding those domains that resolve to a very small set of IPs, and at the same time are characterized by a very short TTL. This rule has been introduced because some attackers setup their flux domains to resolve to only one flux agent (i.e., one IP address) per query. However, in this case the TTL is often set to zero (or at most a few seconds) to guarantee that the flux agent's IP is not cached by the local DNS resolver for too long, and the next time a user "clicks" on the same domain she will obtain a "fresh" flux agent IP, thus making sure the user will be redirected to a reachable compromised machine.

Summing up, the output of the *Message Pre-Filtering* module is a list of candidate flux domains, and their related aggregated DNS information (i.e., resolved IP addresses, average TTL, etc.).

3.5 Domain Clustering

At the end of each epoch E of one day, we consider the list of all candidate flux domains output by the *Message Pre-Filtering* module along with the related DNS information collected during that day, and then we group the domains according to similarities in their resolved IP sets. This clustering step is motivated by the following reasons. Botmasters usually operate malicious flux services using a (often large) number of fast-flux domain names that all point to flux agents related to the same flux network. We speculate that one of the reasons for this behavior is to evade domain blacklists. During our study, we came across a number of malicious flux services advertised through large sets of "random-looking" domain names. The botmaster seemed to be registering many new domain names every day to compensate for older domain names that were identified as malicious by security operators and added to blacklists.

Our clustering approach groups together domain names that within an epoch E resolved to a common set of IP addresses. To perform domain clustering of flux domains that are related to each other, we use *single-linkage* hierarchical clustering algorithm [14], [15], which adopts a "friends of friends" clustering strategy. In order to apply the clustering algorithm to a set of domain names $\mathcal{D} = \{d_1, d_2, ..d_n\}$, we first need to formally define a measure of similarity between them.

We define the following similarity metric between pairs of candidate flux domains. Given two domains α and β, and their cumulative set of resolved IP addresses collected during an epoch E, respectively R_α and R_β, we compute their similarity score as

$$sim(\alpha, \beta) = \frac{|R_\alpha \cap R_\beta|}{|R_\alpha \cup R_\beta|} \cdot \frac{1}{1 + e^{\gamma - \min(|R_\alpha|, |R_\beta|)}} \in [0, 1] \quad (1)$$

The first factor is the *Jaccard index* for sets R_α and R_β, which intuitively measures the relative overlap between the two cumulative sets of resolved IPs. The second factor is a sigmoidal weight designed to measure the confidence in estimating the Jaccard index. Let us recall that for each domain d we are able to collect a *sample* of the IP addresses that domain is mapped to (depending on the number of DNS messages about d contributed to ISC/SIE). Thus, in some cases the aggregated message $Q(d)$ may contain a small number of IP addresses. When the size of sets R_α and R_β is small, the Jaccard index may still have a large value if the sets exhibit a large intersection, but this value can be affected by uncertainty due to the sampling effect. Therefore, we translate this uncertainty into a weight that reduces our confidence on the Jaccard index, when the resolved IP sets are small. To better understand the choice of this weight factor consider this example: if $|R_\alpha \cap R_\beta| = 3$ and $|R_\alpha \cup R_\beta| = 4$ or $|R_\alpha \cap R_\beta| = 30$ and $|R_\alpha \cup R_\beta| = 40$, the Jaccard index is 0.75 in both cases, but the similarity measure related to the second case should be larger than the one related to the first case. The parameter γ is chosen a priori, according to our experience in considering a set of resolved IPs as being "small" or "large". In particular, we found that the value of $\gamma = 3$ worked well in our experimental evaluation. Thus it follows that if $\min(|R_\alpha|, |R_\beta|) = 3$ the weight factor will be equal to 0.5. For larger numbers of resolved IP addresses, the sigmoidal weight tends to its asymptotic value of 1 (e.g., when $\min(|R_\alpha|, |R_\beta|) = 10$, the weight factor is 0.999).

Using the similarity measure defined above, we can compute a similarity (or proximity) matrix $\mathsf{P} = \{s_{ij}\}_{i,j=1..n}$ that consists of similarities $s_{ij} = sim(d_i, d_j)$

between each pair of domains (d_i, d_j), and then we can apply the hierarchical clustering algorithm. At the beginning of the algorithm, each domain is considered as a cluster. Then, at each step, the two nearest clusters are merged. The algorithm stops when all the domains are included in one cluster. The obtained sequence of nested clusters is usually represented as a *dendrogram*, i.e., a tree-like data structure in which the leaves represent the original domains in $\mathcal{D}$, and the length of the edges represent the distance between (sub-)clusters [14].

The obtained *dendrogram* does not actually define a partitioning of the domains into clusters, rather it defines a hierarchy of "relationships" among domains. A partitioning of the set $\mathcal{D}$ into clusters can then be obtained by cutting the dendrogram at a certain height h. The leafs that form a connected sub-graph after the cut are considered as part of the same cluster [14]. Of course, different values of the height of the cut h may produce different clustering results. In order to choose the best dendrogram cut (i.e., the best clustering), we apply a clustering validation approach based on *plateau* regions [16], similar to the approach we used in [7]. In practice, we plot a graph that shows how the number of clusters varies by choosing different values of h, and we look for *plateau* (i.e., flat) regions in the graph that are an indication of "stability" or *natural clustering*. Plateau regions correspond to those steps of the algorithm where the two nearest clusters that have to be merged exhibit a quite low measure of similarity. In our experiments we found that a cut threshold of $h = 0.75$ yielded an effective clustering of the domains. We also noted that FluxBuster's results are not very sensitive to this threshold, as varying the value of h within a relatively wide range around $h = 0.75$ has a negligible impact on the clustering, and the classification results. We will discuss the clustering results more in detail in Section 4.

Summing up, the output of the *Domain Clustering* module is a set $\mathbf{C} = \{\mathcal{C}_i\}_{i=1..l}$ of clusters of candidate flux domains, where domains belonging to the same clusters are related to each other, and each cluster $\mathcal{C}_i$ represents a candidate flux network. As discussed, in Section 3.1, each cluster $\mathcal{C}_i$ for which the size of the resolved IP $|R_i| > \kappa$ (e.g., $\kappa = 30$) will then be sent to the *Classifier* module.

3.6 Statistical Features

In order to classify a candidate flux network (i.e., a domain cluster) $\mathcal{C}$, we need to describe $\mathcal{C}$ in terms of a feature vector that can be processed by the statistical classifier employed in the *Classifier* module, as discussed in Section 3.1. In [4], Passerini et al. proposed a thorough characterization of fast-flux domain names in terms of statistical features for supervised learning, and introduced a set of nine features based on the analysis of the set of IP addresses resolved by *actively querying* single domain names. In this work we adapt some of the features proposed in [4] to characterize clusters of domain names (as opposed to single domains) related to malicious flux networks, and we introduce several additional new features that are more suitable to the *passive analysis* we perform on the ISC/SIE [8] DNS data.

First, we will provide a definition of our features, and then we will motivate our choices. In the following, we take as a reference a generic domain cluster $\mathcal{C}$ computed at the end of an epoch E_m, and we assume that R represents the set of all distinct resolved IP addresses collected during epoch E_m that are related to the domains in $\mathcal{C}$.

- ϕ_1 **Number of resolved IPs**. Overall number of distinct resolved IP addresses in R.
- ϕ_2 **Number of domains**. Total number of distinct domain names in the cluster.
- ϕ_3 **Avg. TTL per domain**. The average TTL of the domains in the cluster[3].
- ϕ_4 **Number of domains per network**. Number of distinct domain names observed during the previous epochs, E_{m-1}, E_{m-2}, ..., E_{m-N}, that share at least one resolved IP with the domains in $\mathcal{C}$ (in our experiments we set $N = 5$ epochs).
- ϕ_5 **IP diversity**. Normalized entropy of the `/16` network prefixes (see Section 3.4) of the IPs in R, computed as follows:
$$\phi_5 = \frac{-\sum_x p(x) \cdot \log_2 p(x)}{\log_2(\phi_1)}$$
where the probability $p(x)$ is given by the relative frequency of the network prefix x.
- ϕ_6 **IP Growth Ratio**. Average number of *new* IP addresses "discovered" during E_m per each DNS query related to domains in $\mathcal{C}$.
- ϕ_7 **IP Last Growth Ratio** (2 features). Average number of *new* IP addresses per DNS query, as discovered by analyzing the *last* de-duplicated DNS message associated to each domain in $\mathcal{C}$ during E_m. We compute two versions of this feature. In one case, this feature is computed independently for each domain in $\mathcal{C}$, and its average value is selected (ϕ_{7a}). In the other case, this feature is computed by analyzing the last de-duplicated message among all the messages related to domains in $\mathcal{C}$ (ϕ_{7b}).
- ϕ_8 **IP Prefixes Last Growth Ratio** (2 features). This feature is similar to ϕ_7, but it is based on *new* IP prefixes. In practice, we compute the average number of `/16` network prefixes per DNS query, as discovered by analyzing the last de-duplicated DNS message associated to each domain in $\mathcal{C}$ during E_m. Similarly to ϕ_7, we compute two versions of this feature. One is computed independently for each domain in $\mathcal{C}$, and its average value is selected (ϕ_{8a}), while the other is computed by analyzing the last de-duplicated DNS message among all messages related to domains in $\mathcal{C}$ (ϕ_{8b}).

3. To be more precise, this feature measures the average TTL of the resource records of type `A` related to the domains in the cluster.

ϕ_9 **Novelty** (3 features). Consider the set R of resolved IP addresses observed during epoch E_m for the domains in cluster $\mathcal{C}$. To compute the **Novelty** features, we go back in time and look at the overall set of IPs pointed by domains in $\mathcal{C}$ during W previous epochs (E_{m-W}, E_{m-W+1}, ..., E_{m-1}). Let this set of IP addresses be R'. Then, we compute the number of *new* IP addresses, i.e., IPs that belong to R but not to R', and we divide this number by the number of epochs since DNS messages related to domains in $\mathcal{C}$ were observed for the first time. In our experiments, we computed ϕ_9 for three different values of W, that is, $W = 7$ (ϕ_{9a}), $W = 30$ (ϕ_{9b}), $W = 180$ (ϕ_{9c}).

Motivation: The features defined above are designed to capture the characteristics of flux networks discussed in Section 3.3. While the relationship between the description of the characteristics in Section 3.3 and features ϕ_1 to ϕ_5 is straightforward, some additional reasoning is needed to understand the motivations behind features ϕ_6 to ϕ_9. These features aim to better estimate how the set of resolved IPs (and their network prefixes) for domains in $\mathcal{C}$ grows in time. The larger the value of these features, the higher the likelihood that the set of IP addresses R is changing rapidly. Therefore, high values for these features provide strong support that $\mathcal{C}$ is related to a flux network (see Section 3.3).

It is worth noting that features ϕ_5 and ϕ_8 are based on `/16` IP prefixes. The reason why we choose `/16` prefixes is because they tend to approximate network boundaries fairly well. In other words, if two IP addresses have different `/16` prefixes, it is very likely that they belong to different organizations (or two different subnetworks within a large organization). While mapping IP addresses to their autonomous system (AS) numbers or BGP prefixes may give us a more accurate estimate of whether two IPs belong to different networks, we observed that `/16` prefixes effectively approximate this information without the computational burden required to compute *IP-to-AS* or *IP-to-BGP prefix* mappings. The rationale behind these features (e.g., ϕ_5) is that, unlike in the case of CDNs or other legitimate services, flux agents are often scattered across many networks located in many different countries, thus increasing the number of IP addresses that do not share a common `/16` prefix (see Section 3.3).

It is worth noting that while features ϕ_7,ϕ_8, and ϕ_9 may appear somewhat correlated, they all offer different pieces of information that contribute to improving classification accuracy (this is supported by an analysis of the decision tree obtained by training the classifier described in Section 3.7).

3.7 Flux Classifier

Each cluster $\mathcal{C}_i$ can be seen as a *candidate flux network* defined by the set of all the domain names in $\mathcal{C}_i$, and the overall set of IP addresses these domains resolved to during an epoch E. At the end of each epoch E, and for each cluster $\mathcal{C}_i$, we measure the features described in Section 3.6, and employ the popular C4.5 decision-tree classifier [17] to automatically classify a cluster $\mathcal{C}_i$ as either *malicious flux network* or *legitimate/non-flux network*. The reasons for using a decision-tree classifier are as follows: a) decision-trees are efficient and have been shown to be accurate in a variety of classification tasks; b) they output a set of classification rules, which allow us to asses what features are the most effective in detecting malicious flux networks; c) C4.5 is able to automatically prune those features that are not useful or that introduce noise, rather than increasing classification accuracy [17]. We first train the C4.5 classifier on a *training dataset* containing a number of labeled clusters related to malicious flux services and a number of clusters related to legitimate/non-flux services. Afterwards, the trained classifier is used to classify the clusters obtained at the end of each epoch E from the ISC/SIE [8] data, as shown in Figure 2. The details on how we obtained the labeled training dataset and evaluated the accuracy of the *Classifier* module are reported in Section 4.

4 Evaluation

In this section we discuss the results of our evaluation of FluxBuster.

4.1 Experimental Setup

To train and evaluate our detection system, overall we used about 10 months of data collected through the ISC Security Information Exchange[4] from June 2010 to March 2011. We used about four months of data (from June 2010 to September 2010) to build a *labeled dataset*, which we will refer to as **LDS**. We used **LDS** for two purposes: (1) for estimating the accuracy of the *Classifier* module through 10-fold cross validation; and (2) to train FluxBuster's *Classifier* module before deployment. After training, we used approximately one additional month of data for a preliminary validation of the system and parameter tuning, and finally we deployed and evaluated FluxBuster over the remaining (almost) five months (from the 24th of October 2010 to the 15th of March 2011).

In the following, whenever we mention domain names we refer to *second-level* domains (or 2LDs, for short), unless otherwise specified. To give an easy definition of 2LD, consider the following example: given the domain name `news.l.google.com`, we refer to `.com` as the *top-level* domain (TLD), `google.com` is the 2LD, while we refer to the entire domain string as the *fully qualified* domain (FQD). This definition can be generalized to any domain name, e.g., `c.b.a`, where `a` is the TLD, `b` is the 2LD, etc. In practice, identifying the "real" TLD is not always straightforward. For example, *delegation only* domains such as `co.uk` act as *effective TLDs*. Therefore,

4. We harvested DNS messages from *channel 204*.

given for example the domain www.bbc.co.uk, co.uk is the effective TLD, and bbc.co.uk is the 2LD. To accurately identify effective TLDs we make use of the Mozilla Public Suffix List[5].

4.1.1 Labeled Dataset

To obtain the labeled dataset **LDS** we used a semi-manual process. We built a user-friendly web interface that allowed us to quickly sort domain clusters according to different characteristics (e.g., number of domains, number of IPs, IP diversity, etc.), verify whether (and what kind of) a website is served under a given domain, and manually label the domain clusters as either *flux* or *non-flux*. We partially automated the labeling process by making extensive use of prior information regarding known flux domains, known malware domains, and legitimate popular domains (see Section 4.1.2). To reduce the number of errors (i.e., the noise) in the labeled dataset, whenever a clear cut decision between *flux* or *non-flux* clusters was not possible (not even after extensive manual analysis), we marked those domain clusters as "unknown" and excluded them from the dataset. Overall, we were able to label 1,337 domain clusters as *flux* and 5,708 as *non-flux*, which overall included 2,116 distinct 2LDs (59,215 FQDs) and 100,644 distinct 2LDs (113,580 FQDs), respectively. We labeled 313 clusters as "unknown". We noticed that many of these clusters were suspicious, but could not find sufficient information to label them as certainly *flux*.

4.1.2 Evaluation Data and Ground Truth

Evaluating a detection system such as FluxBuster in a live operational environment is a challenging task. The main reason is that it is not possible to obtain complete ground truth on the nature of the classified data. For example, given a domain cluster $\mathcal{C}$ classified by FluxBuster as *flux*, $\mathcal{C}$ may fall into three categories: (1) $\mathcal{C}$ includes domains and/or IPs that are known to be related to a flux network (in which case this is a true positive); (2) $\mathcal{C}$ does not represent a flux network, and may instead represent a CDN or other legitimate services (in which case we have a false positive); (3) the true nature of $\mathcal{C}$ is *unknown*, that is no prior information exists on this cluster in any public (or even private) security data sources. In practice, the case when FluxBuster correctly detects a previously unknown flux network is in a way analogous to discovering a *zero-day* malware or exploit *in-the-wild*, whereby confirming the correctness of the classification usually requires manual analysis.

The main implication of the existence of these *unknown* cases that cannot be easily confirmed (if not with extensive forensic analysis), is that in the live evaluation we cannot limit ourselves to measuring the *true positive* (TP) rate and *false positive* (FP) rate, because other quantities such as the *false negative* (FN) rate and *true negative* (TN) rate cannot be directly derived from the TP rate and FP rate[6], respectively. Therefore, in our evaluation we compute the true positives, false positives, false negatives, and true negatives separately, using data labeled by leveraging public information about legitimate, *flux*, and malware-related domain names, as discussed below.

To evaluate the true positives of our classification system, we gathered two datasets of *blacklisted* domains: (i) a dataset of known flux domains, which we will refer to as the **KFD** dataset; and (ii) a dataset containing known malware-related domain names, which we refer to as the **KMD** dataset.

The **KFD** dataset contained 75 domain names (2LDs) classified as flux and reported by the abuse.ch website, which is well known and considered as a trusted source by many security professionals. The **KMD** contained 26,496 domains that were collected automatically and updated daily by harvesting public domain blacklists from twelve different reputable sources, including malwaredomains.com, malwarepatrol.com, etc. We started to collect the data included in the **KFD** and **KMD** datasets at the end of August 2010, and since then we have kept them up-to-date with daily updates. The rationale behind using the **KMD** is that although this dataset contains many domain names that are not related to flux networks, if a malware domain m passes the pre-filtering stage (thus becoming a candidate flux domain) and also FluxBuster flags a domain cluster that includes m as having the characteristics of a flux network, it is extremely likely that the cluster is indeed a flux network (we manually confirmed a large number of these cases).

To evaluate the *false positives*, we collected three datasets of *whitelisted* domains: (i) a dataset containing the top 100,000 domain names according to Alexa.com, which we refer as the **ATD** dataset; (ii) a dataset containing 304,248 distinct second-level domains (321,411 fully qualified domain names) drawn at random from the Yahoo! DMOZ project (which lists manually verified websites), referred to as **YDD** dataset; and (iii) a manually compiled **CDN** dataset containing a list of 31 second-level domains (2LDs) related to both well known and less well known legitimate CDNs. The **ATD** dataset actually contained 57,910 domains that are *consistently top*, i.e., those domains that remained within the top 100k ranking for almost one entire year. We followed the procedure described above to filter out possible noise caused by potentially malicious domains that may have become popular for a short amount of time. The assumption is that the consistently top domains are legitimate, since malicious domains typically have a short life span (this is particularly true for the popular ones that attract the attention of the security community).

We were also interested in computing the *false negatives* and *true negatives* of our classifier. To compute the false negatives we again leveraged the **KFD** and **KMD** datasets. In practice, if FluxBuster classifies a domain

5. http://publicsuffix.org/list/

6. When testing on a perfectly labeled dataset we can always compute $FN = 1 - TP$, and $TN = 1 - FP$.

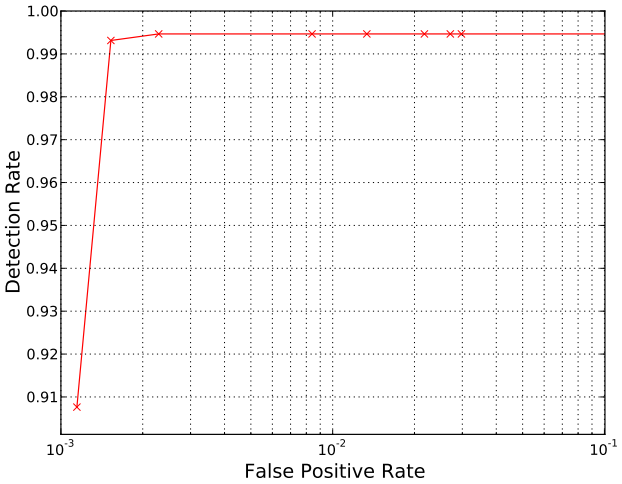

Fig. 3. Receiver Operating Curve for the decision tree classifier, evaluated on the **LDS** dataset (AUC=0.994).

cluster $\mathcal{C}$ as *non-flux* while the domains in the cluster belong to **KFD** or **KMD**, we consider this to be a false negative. It is worth noting that this approach may overestimate FluxBuster's false negatives, because most of the domains in **KMD** are not flux domains. However, since FluxBuster's *Classifier* module only receives candidate flux networks from the previous modules (see Section 3.1), again we assume that if a malware domain belongs to a candidate flux network there is a high probability that the domain is actually *flux*. On the other hand, to compute the true negatives we leveraged the **ATD**, **YDD**, and **CDN** datasets. In this case things are more straightforward: domains from these three datasets are highly likely legitimate, and therefore if FluxBuster classifies a domain cluster containing any of these domains as *non-flux*, we assume the classifier was indeed correct.

4.2 Experimental Results

We now present the results of our evaluation both on a completely labeled dataset and on data collected during an 5-month operational deployment of FluxBuster at ISC/SIE [8].

4.2.1 Cross-Validation

Using the labeled dataset **LDS** described in Section 4.1.1, we performed 10-fold cross validation to estimate the accuracy of the *Classifier* module. The decision tree classifier was able to achieve 99.3% detection rate at 0.15% false positive rate, with an area under the ROC curve (AUC) of 0.994 (see Figure 3). These results confirm that FluxBuster can detect malicious flux networks with high accuracy.

4.2.2 Live Evaluation

Along with the 10-fold cross validation on the labeled dataset **LDS**, we evaluated our detection system in a real-world deployment scenario. To this end, we first trained FluxBuster on the entire **LDS** dataset, and then deployed it at ISC/SIE. Overall, in a period of about 5 months of operational deployment, FluxBuster classified 4,084 domain clusters as flux and 3,633 domain clusters as non-flux, which included a total of 1,743 2LDs (63,442 FQDs) and 227,667 2LDs (264,550 FQDs), respectively. These results were obtained for a threshold $\kappa = 30$ (see Section 3.1). Table 1 shows how these 1,743 2LDs were distributed across different TLDs. For comparison reasons, we also report the distribution of TLDs for the domains in the **KFD** dataset.

Count	TLD
1115	com
377	info
180	net
21	ru
17	org
12	biz
6	me
15	*others*

Count	TLD
38	com
15	ru
14	net
3	cn
2	info
1	tk
1	im
1	cc

TABLE 1
Distribution of TLDs for flux domains detected by FluxBuster (left) and for domains in **KFD** (right).

We now discuss how we separately measured the number of TP, FP, FN, and TN (the choice of separately measuring all these quantities is motivated in Section 4.1.2).

True Positives: We measured true positives in two ways:

- *Known Flux*. We computed the number of domains classified as flux by FluxBuster that were also in the **KFD** dataset. We found that FluxBuster correctly classified 24 out of 75 **KFD** domains (2LDs) as flux. The remaining 51 domains were not classified by FluxBuster, simply because they were not visible from the SIE data feed. To be more precise, 12 of these 51 domains appeared in the SIE data feed, but only very rarely, and they were associated with very few (usually less than five) resolved IPs. In practice, these domains were not considered by FluxBuster, simply because the DNS activity seen for these domains was too low (i.e., very few users were "clicking" on them).
  We also computed how many new flux domains we were able to detect, using the **KFD** domains as a "seed". In practice, starting from **KFD**, we first found the set FC of domain clusters classified by FluxBuster as flux that contained at least a domain $d \in$ **KFD**. Then, we computed how many of the domains in these FC clusters were missing from **KFD**. Using this approach, we were able to find a total of 525 new flux domains that belong to a malicious flux network also pointed by domains in **KFD**. Furthermore, we found that of these 525 new flux domains, 79 of them were known malware domains belonging to the **KMD** dataset. Following

a natural *guilty by association rule*, the remaining 446 can also be regarded as new malicious flux domains (which we also manually verified). These previously unknown malicious flux domains were discovered by FluxBuster thanks to our large-scale passive analysis approach.

- *Known Malware*. We found that a total of 179 domains (2LDs) in the **KMD** dataset were classified as flux by FluxBuster. It is worth noting that while these 179 domains represent only a small fraction of the **KMD** dataset, this dataset contains generic malware domains, some of which are flux domains. However, the vast majority of domains in **KMD** are not necessarily related to flux networks and are simply used for other malicious purposes. In fact, our SIE feed contained 10,447 domains pertaining to the **KMD** dataset, but most of them were been discarded in the pre-filtering stage (Section 3.4), because they do not exhibit the characteristics of flux domains.
  Similarly to the *Known Flux* experiments described above, we then used the **KMD** domains as a "seed" to find clusters of domains that were classified as flux by FluxBuster and that contained at least one malware domain. Doing so allowed us to discover 549 new flux domains representing previously unknown malware domains, which can be classified as malware-related using again a *guilty by association* rule (because they point to the same flux networks pointed by known malware domains).

False Positives: We measured the number of false positives as follows. We first checked how many domains classified as flux by FluxBuster were also present in the **ATD** dataset. We found that only 2 fully qualified domains out of the 57,910 2LDs in **ATD** were classified as flux. These two domains were `pool.ntp.org` and `gyq3606.meibu.com` (both `ntp.org` and `meibu.com` were present in **ATD**). The misclassification of pool.ntp.org was actually expected, because subdomains of `ntp.org` are often used to point to large sets of NTP time servers around the world. For example, in only one day the domain `pool.ntp.org` resolved into 323 different IP addresses scattered around the Internet, thus resembling the behavior of an "aggressive" flux domain (the average TTL was 270 seconds). On the other hand, `gyq3606.meibu.com` was a "true misclassification". On 2010-12-26, `gyq3606.meibu.com` resolved into 75 different IPs, with a TTL of 10 seconds. Although the IP diversity was low (around 0.5), FluxBuster mistakenly assigned the *flux* label.

Afterwards, we checked how many domains classified as flux were part of the **CDN** dataset. Over the entire evaluation period (almost five months), we found that only 35 fully qualified domains under 4 distinct 2LDs caused false positives. These four 2LDs were related to small CDNs, precisely `swiftcdn1.com`, `cloudex.net`, `rncdn1.com`, and `nefficient.co.kr`. We noticed that in all these cases, the clusters shared most features with flux domains, but the IP diversity was always quite low, only slightly above 0.5. We found this was a "defect" of the decision tree learned by FluxBuster during training, and therefore we manually added a rule whereby no cluster with IP diversity lower than 0.6 should be classified as flux. This simple rule filtered out all the false positives without any change at all to the number of true positives.

We also checked the flux domains against the **YDD** dataset, and we found no false positives in this case.

True Negatives: Overall, FluxBuster classified a total of 3,633 domain clusters as non-flux, which included a total of 227,667 2LDs (264,550 FQDs). We noticed that many clusters contained large numbers of legitimate domains that shared the same set of IPs because they all shared the same physical web-server. For example, we found large clusters of personal websites that appeared to rely on *virtual hosting* services.

We checked the 227,667 2LDs against our datasets of legitimate domains (**ATD**, **YDD**, and **CDN**), and we found that 171 2LDs were confirmed to be non-flux. We could not find an automatic way to confirm the remaining *unknown* domains (this is why we compute both true negatives and false negatives, as discussed in Section 4.1.2).

False Negatives: To measure the false negatives, we checked how many of the domains belonging to the **KFD** dataset were consistently classified by FluxBuster as non-flux. We found that only one such domain belonged to **KFD**: `discountpharmacyhealth.net`. This false negative was due to the fact that `discountpharmacyhealth.net` resolved into a relatively low number of distinct IP addresses (less than 60 overall), had a low value of the novelty features, and a relatively low IP diversity. Similarly, we computed how many of the domains consistently classified as non-flux were in the **KMD** dataset. We found 30 such domains (out of the 26,496 domains in the dataset).

It is worth noting again that **KMD** is not perfect. In fact, it contained some legitimate domains resulting from the misclassification of the third-party systems used to create the malware domain blacklist (we manually pruned obviously legitimate domains from **KMD**). However, the fraction of noisy (i.e., non-malware) domains is very low, and we therefore decided to use this dataset, although it may slightly overestimate our false negatives.

4.2.3 Early Detection Results

As we mentioned above, among the domains classified as flux by FluxBuster, 24 of them appeared in **KFD**, and 179 appeared in **KMD**. Of these domains that were confirmed to be malicious, we were interested in knowing how many of them we detected earlier then when they were detected by third-party security services, and how much earlier. Overall, we found that FluxBuster detected 9 out of 24 confirmed flux domains earlier than when they appeared in the **KFD** dataset (the remaining

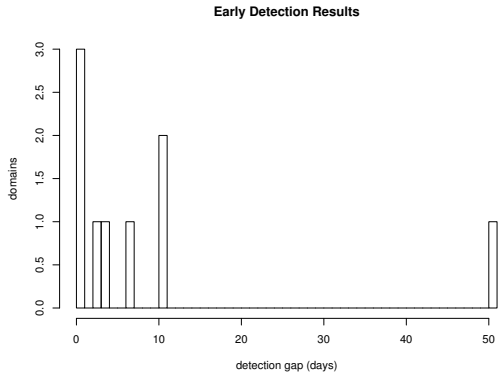

Fig. 4. Early detection of **KFD** domains.

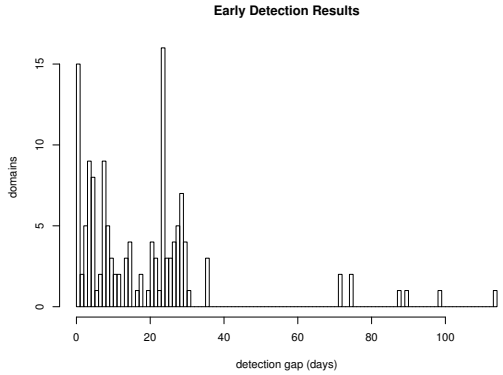

Fig. 5. Early detection of **KMD** domains.

15 domains were detected later than they appeared in **KFD**), and 125 out of 179 confirmed malware domains earlier than when they appeared in **KMD**. To gain a better view of how earlier FluxBuster was able to detect these malicious domains, we proceeded as follows. Let d be a flux domain, t_{FB} be the day when d was detected by FluxBuster, t_{KFD} be the day when d appeared in **KFD**, and t_{KMD} be the day when d appeared in **KMD**. Figure 4 shows how many domains we were able to detect for each value of $\delta_{KFD} = (t_{KFD} - t_{FB})$, while Figure 5 shows how many domains we were able to detect for each value of $\delta_{KMD} = (t_{KMD} - t_{FB})$. As the graphs show, FluxBuster is often able to detect malicious flux domains days or even weeks before they are detected by well-known third-party security services. In addition, we isolated domain names related to the infamous *Zeus botnet* [18] from the **KMD** dataset. We found that FluxBuster detected 13 out of 21 Zeus domains earlier then when they appeared in **KMD**. Figure 6 shows the distribution of early detection days for these domains.

4.2.4 IP-based analysis

In addition to a domain-based evaluation, we were interested in measuring the effectiveness of FluxBuster in terms of its ability to enumerate the flux agents (i.e., IP addresses resolved by domains pertaining to flux networks). In the following, we present some interesting results that highlight the usefulness of FluxBuster with respect to other state-of-the-art systems (which are mainly based on active-probing, rather than passive DNS monitoring) such as the *flux-tracker* at `abuse.ch`.

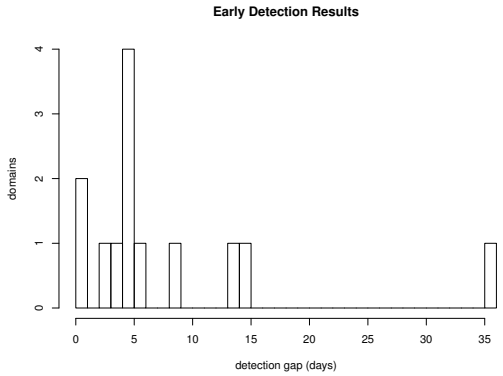

Fig. 6. Early detection of Zeus botnet domains.

Using the on-line DNSBL lookup service[7] provided by `abuse.ch`, we measured the fraction of flux agents (i.e., IPs) enumerated by FluxBuster that were known or not to `abuse.ch`. In particular, for each epoch, at the end of FluxBuster's classification process, we randomly sample 10 flux agent IPs from the clusters labeled as flux, and we query `abuse.ch`'s DNSBL on this IPs. It is worth noting that we only consider flux clusters that contain at least one domain name that is also present in **KFD** (i.e., the dataset of known "top" flux domains reported by `abuse.ch` itself). Figure 7 shows, per day, the fraction of IPs (i.e., flux agents) enumerated by FluxBuster that were listed as malicious by `abuse.ch`. We found that, in average, about 62% of the tested IPs were actually unknown to `abuse.ch`. This suggests again that FluxBuster can offer a valuable complementary view of flux networks in terms of flux domain names and their flux agents.

We were also interested in evaluating, using an IP-based analysis, the number of new, previously unknown flux domains can be "confirmed" using again a simple *guilty-by-association* rule. To this end, we first considered (per each day) the set P of IP addresses (i.e., flux agents) contained in flux clusters that included at least one domain name belonging to the **KFD** or **KMD** datasets. Then, we retrieved (per each day) the set of all the domain clusters $C = \{C_1, C_2, ..., Cn\}$ that were classified as *flux* by FluxBuster and that contained at least one domain $p \in P$. Finally, we counted the number of distinct single domain names contained in the flux clusters in C that were not present in **KFD** or **KMD**. We found 1,030 such domains, and we verified that they were mostly related to pharmacy scams and porn-related websites, and that none of these domains appeared in **ATD**, **YDD**, and **CDN**. In practice, through this process we were able to confirm the fact that FluxBuster was able to discover

7. http://dnsbl.abuse.ch/faq.php

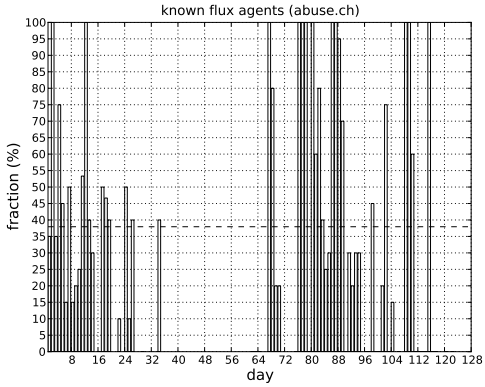

Fig. 7. Fraction of flux agents among those detected by FluxBuster that are also reported by the DNSBL at `abuse.ch` (only flux clusters that include at least one domain belonging to **KFD** are considered). The dotted line shows the average value.

1,030 previously unknown malicious flux domain names.

4.2.5 Flux Websites

In our experiments we found that the vast majority of flux domains detected by FluxBuster resolve to flux agents that successfully respond to HTTP requests (on default port 80) and return malicious content/scam pages. Such domains represent a threat to web users. To reduce these threats, FluxBuster could provide an interface similar to the Google Safebrowsing API[8], which can be queried by a browser plugin to inform users that they are about to visit a website under a flux domain.

To have a sense of the potential benefits of using FluxBuster under this browser-plugin setting, we counted the number of effective 2LDs that we classified as flux were hosting web content. Figure 8 shows the results obtained over almost one month of data (from April 14 to May 9, 2011). As a point of reference, we also counted how many of these flux websites were flagged by Google Safebrowsing, or were reported as known flux or as malware-related in third-party public blacklists, as explained below.

The *Web-related Flux Domains* represent domain names for which some web content was successfully downloaded. To perform this measurement, for each fast flux domain name, we issued a number of HTTP `GET` requests on the "root" URL `/`, and on other related URLs obtained by searching for the domain name on search engines[9]. The *Known Malware Flux Domains* represent the number of flux domains that also appear in either **KFD** or **KMD**. To compute the *Analyzed by Safebrowsing* domains we queried each flux domain against the Google Safebrowsing service, and we counted the number of domains that Google Safebrowsing reported as *previously tested*. On the other hand, the *Flagged Malicious by Safebrowsing* domains are the ones that have been tested by Google Safebrowsing, and have been labeled as malicious.

It is worth noting that we verified that the vast majority of the flux websites that are not reported by Google Safebrowsing or public blacklists are related to rogue pharmacies, suspicious porn-related websites promoted through thousands of random-looking domains, and a number of other scams. Since Google Safebrowsing, according to the official Google site, aims to report only "suspected phishing and malware pages", this large difference in number of domains reported by FluxBuster and the ones flagged as malicious by Google Safebrowsing was expected.

We would like to emphasize that this analysis aimed to show that FluxBuster can complement other public *web security* services by providing additional useful information to users who prefer to avoid visiting any kind of potentially malicious websites (including rogue pharmacies, suspicious porn-related sites, and other scam websites), while browsing the Web.

8. http://code.google.com/apis/safebrowsing/

9. For example, for each domain we queried for `inurl:<domain name>` on Google to find related URLs.

Flux domains	**1,743** 2LDs (**63,442** FQDs)
Flux agent IPs	**317,203** distinct IP addresses (on average 3,265 distinct IPs per day)
Previously unknown flux 2LDs	**995** through a "domain-based" analysis, and **1,030** through an "IP-based analysis" (using *guilty-by-association*)
Early-detection results	**64.5%** of malicious 2LDs detected earlier than other state-of-the-art tools (131 2LDs out of 203)
Previously unknown flux agent IPs	**62%** of flux agents tested against `abuse.ch` DNSBL service

TABLE 2
Summary of some of the results obtained by FluxBuster during the *live* evaluation.

5 Discussion and Future Work

Table 2 summarizes some of the results of the evaluation of FluxBuster in a real-world deployment. It is worth noting the number of distinct flux agent IPs is not necessarily indicative of the exact number of flux agents. As shown in [19], the number of distinct IPs is an upper bound on the number of infected machines, mainly due to the effect of DHCP churn.

The evaluation results show that FluxBuster is able to detect flux networks *in-the-wild*, and can often do so several days or even weeks earlier than other state-of-the-art detection systems. This does not mean, though, that FluxBuster should replace other flux and malware detection systems. Instead, we designed FluxBuster to complement other detection systems, and, in particular, to compensate for some of the limitations of flux detection systems based on active probing.

FluxBuster has its own limitations. For example, for a flux domain to be detected, FluxBuster needs to observe at a minimum one DNS message related to that

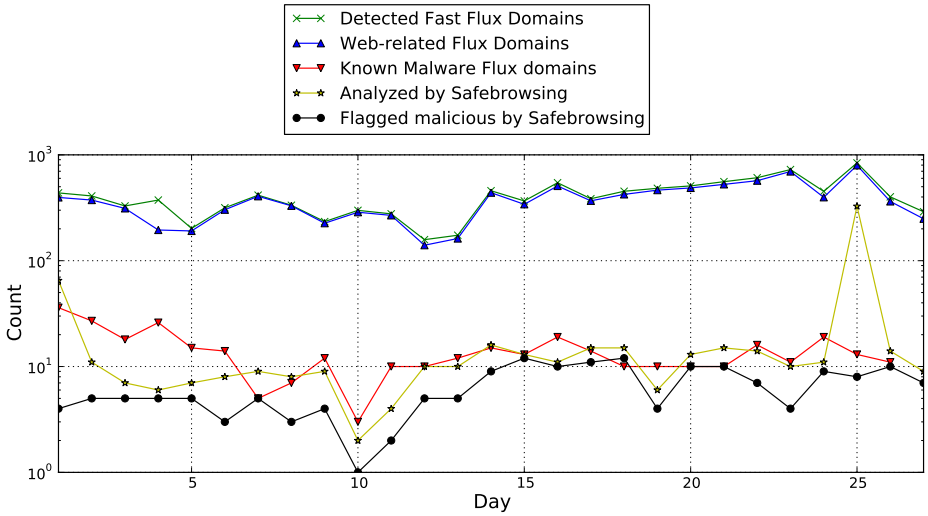

Fig. 8. Analysis of flux domains that host web content.

domains. Because FluxBuster's observations depend on users' behavior (e.g., "clicking" on domain names), if a flux domain is never queried by any of the users within the monitored networks, FluxBuster will not be able to detect it. Also, enough IP addresses (see the $\kappa = 30$ threshold in Section 3) need to be collected during a given epoch (e.g., one day) to perform the classification of a domain cluster. This may cause some false negatives, as discussed in Section 4.2.2. However, FluxBuster aims to detect flux domains, and more in general flux networks, that are active and successfully attract victim users in at least one epoch.

Our goal is to detect these malicious domains as early as possible, and help to prevent other users from falling victim of the same threats. However, currently we can only detect new flux domains at the end of each epoch (i.e., one day). In the current deployment, it may take up to 30 hours for FluxBuster to generated the detection results (24 hours for traffic monitoring, plus up to 8 hours for processing). In our future work, we intend to study how to shorten FluxBuster's response time.

As mentioned in Section 4.2.2, our live evaluation is limited by the difficulties in collecting perfect ground truth for all the domains processed by FluxBuster during the five-month real-world deployment. Nonetheless, we were able to verify a significant fraction of the classification results. The live evaluation results, as well as the cross-validation results reported in Section 4.2.1, confirm that FluxBuster can detect malicious flux networks with very low false positives.

While FluxBuster detects flux domains in a *stealthy* way, thanks to the use of purely passive DNS traffic monitoring, an adversary who learns how FluxBuster works may attempt to modify the way her flux network operates and the served malicious content is promoted to try to avoid being detected. In [20], Knysz et al. discuss a number of potential evasion techniques against flux detection systems. However, the proposed evasion strategies mainly focused on evading active-probing-based systems, and may not successfully thwart FluxBuster's detection [20]. One possible way in which an adversary may attempt to evade FluxBuster is to setup her flux domains to introduce noise into the set of resolved IPs. Namely, the adversary may setup the flux domains to resolve to a set of IPs that contains both a number of flux agent IPs as well as some random legitimate IPs mixed in. The effect of this evasion attack is to mislead the domain clustering algorithm, thus potentially affecting the accuracy of the classification process. However, it is worth noting that this strategy may negatively impact the flux network itself. In fact, FluxBuster only analyzes DNS messages deriving from the users' "clicks". If the adversary introduces noise in the resolved IP set, this means that a significant portion of the potential victims may be redirected to legitimate IPs, for example, rather than to the IPs of real flux agents, thus reducing the distribution of malicious content to only a fraction of the users. While more sophisticated evasion strategies may be devised, the passive monitoring approach followed by FluxBuster naturally raises the bar for the adversaries who attempt to evade detection, forcing them to incur a significant cost to change the way they setup and run their flux networks and related cyber-criminal operations.

Currently FluxBuster can process one day worth of DNS traffic collected from hundreds of distributed ISC/SIE sensors in about eight hours. In our future work we plan to make FluxBuster more scalable to better keep pace with the foreseeable growing volume of DNS traffic due to the increase in the number of networks that contribute DNS messages to ISC/SIE.

6 Conclusion

In this paper we presented FluxBuster, a novel system for detecting malicious flux networks *in-the-wild*. FluxBuster

is based on a purely passive, large-scale analysis of DNS messages collected from hundreds of different networks through ISC/SIE. Our long-term evaluation showed that FluxBuster is capable of accurately detecting previously unknown flux networks days or even weeks in advanced before they appear in public blacklists. In addition, we showed that FluxBuster often detects flux domains that may remain unknown to other third-party systems, thus providing valuable information to the security community that can be used to block a variety of malicious content.

ACKNOWLEDGMENTS

The authors would like to thank David Dagon, Wenke Lee, and ISC/SIE for providing access to the DNS data used in this research, and the anonymous reviewers for their helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. OCI-1127195. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was also partly supported by a grant from *Regione Autonoma della Sardegna* awarded to I. Corona, PO Sardegna FSE 2007-2013, L.R.7/2007 "Promotion of the scientific research and technological innovation in Sardinia".

REFERENCES

[1] SSAC, "SAC 025 - SSAC advisory on fast flux hosting and DNS," 2008, http://www.icann.org/en/committees/security/sac025.pdf.
[2] The Honeynet Project, "Know your enemy: Fast-flux service networks," 2007, http://old.honeynet.org/papers/ff/fast-flux.html.
[3] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and detecting fast-flux service networks," in *NDSS '08: Proceedings of the Network & Distributed System Security Symposium*, 2008.
[4] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "Fluxor: Detecting and monitoring fast-flux service networks," in *DIMVA '08: Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008.
[5] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," in *MALWARE '08: Proceedings of the 3rd International Conference on Malicious and Unwanted Software*, 2008.
[6] M. Konte, N. Feamster, and J. Jung, "Dynamics of online scam hosting infrastructure," in *PAM '09: Proc. Passive and Active Measurement Conference*, 2009.
[7] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive dns traces," in *Annual Computer Security Applications Conference (ACSAC)*, 2009.
[8] "ISC security information exchange," https://sie.isc.org.
[9] X. Hu, M. Knysz, and K. G. Shin, "Rb-seeker: Auto-detection of redirection botnets," in *Annual Network & Distributed System Security Symposium (NDSS)*, 2009.
[10] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen, "Fast-flux bot detection in real time," in *Proceedings of the 13th international conference on Recent advances in intrusion detection*, ser. RAID'10, 2010.
[11] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10, 2010.
[12] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Building a dynamic reputation system for dns," in *18th Annual Network and Distributed System Security Symposium*, ser. NDSS, 2011.
[13] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy," in *Proceedings of the 20th USENIX conference on Security*, ser. USENIX Security'11, 2011.
[14] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.
[15] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
[16] R. Dugad and N. Ahuja, "Unsupervised multidimensional hierarchical clustering," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998.
[17] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
[18] "Zeus tracker," https://zeustracker.abuse.ch/.
[19] 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 *Proceedings of the ACM Conference on Computer and Communications Security*, 2009.
[20] M. Knysz, X. Hu, and K. G. Shin, "Good guys vs. bot guise: Mimicry attacks against fast-flux detection systems," in *Proceedings of IEEE INFOCOM*, 2011.

Roberto Perdisci received a M.Sc. degree, with honors, in Electronic Engineering, and a Ph.D. in Computer Engineering at the University of Cagliari, Italy, in 2003 and 2007, respectively. He was also research scholar at the Georgia Tech Information Security Center (2005-2007), and post-doctoral fellow at the College of Computing of the Georgia Institute of Technology (2009-2010). He is currently Assistant Professor in the Computer Science department at the University of Georgia. His main research interests are on network security, and in particular on malware detection at the network level and on DNS security.

Igino Corona received the M.Sc. in Electronic Engineering and Ph.D. in Computer Engineering at the University of Cagliari, Italy, in 2006 and 2010, respectively. In 2009 he also worked at the Georgia Tech Information Security Center as a research scholar. He is currently a post-doctoral researcher at the University of Cagliari. His main research interests include web security, detection of fast flux networks, adversarial machine learning and web intrusion detection.

Giorgio Giacinto is Associate Professor of Computer Engineering at the University of Cagliari, Italy, Pattern Recognition and Applications Group of the Dept. of Electrical and Electronic Engineering (http://prag.diee.unica.it). During his career Prof. Giacinto has published more than seventy papers on international journals, conferences, and books. The main contributions are in the field of "multiple classifier systems", computer security (intrusion detection systems), and content-based image retrieval (relevance feedback techniques). He is associate editor of the Information Fusion journal, and a senior member of the IEEE, and the ACM. He is also a member of the IAPR (International Association for Pattern Recognition).