Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces

Roberto Perdisci[a,b], Igino Corona[c], David Dagon[a], and Wenke Lee[a]
[a]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b]Damballa, Inc. Atlanta, GA 30308, USA
[c]DIEE, University of Cagliari, 09123 Cagliari, ITALY
perdisci@gtisc.gatech.edu, igino.corona@diee.unica.it, {dagon,wenke}@cc.gatech.edu

***Abstract*—In this paper we propose a novel, *passive* approach for detecting and tracking malicious *flux* service networks. Our detection system is based on passive analysis of recursive DNS (RDNS) traffic traces collected from multiple large networks. Contrary to previous work, our approach is not limited to the analysis of suspicious domain names extracted from spam emails or precompiled domain blacklists. Instead, our approach is able to detect malicious flux service networks *in-the-wild*, i.e., as they are accessed by users who fall victims of malicious content advertised through *blog spam*, *instant messaging spam*, *social website spam*, etc., beside email spam. We experiment with the RDNS traffic passively collected at two large ISP networks. Overall, our sensors monitored more than 2.5 billion DNS queries per day from millions of distinct source IPs for a period of 45 days. Our experimental results show that the proposed approach is able to accurately detect malicious flux service networks. Furthermore, we show how our passive detection and tracking of malicious flux service networks may benefit spam filtering applications.**

I. 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* service networks have recently started to thrive [11]. Malicious flux service 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 across 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. Differently from legitimate CDNs, whose nodes are professionally administered machines, the nodes of a malicious flux service network, a.k.a. *flux agents*, are represented by malware-infected machines. The flux-agents are usually part of a *botnet* and can be remotely controlled by the malware author, who is often referred to as the *botmaster*.

Malicious flux service networks are commonly used to host phishing websites, illegal adult content, or serve as malware propagation vectors, for example. The main technical difference between a malicious flux service network and a legitimate CDN is that, while the nodes of a legitimate CDN are highly reliable and tightly controlled by the CDN administrator, botmasters do not 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 redistribute the received content requests accordingly. In order to cope with this problems and maintain high content availability, botmasters usually set up their malicious flux services using *fast-flux* domain names. In practice, fast-flux domain names are characterized by the fact that the set of resolved IP addresses (the flux agents) for these domain names change rapidly, potentially at every DNS query [12]. Furthermore, since it is usually hard for the botmaster to control exactly where the malware propagates and what machines are infected by her *bot* software, the flux agents are often scattered across many different networks [12].

A. Related Work

A number of approaches for detecting fast-flux domain names have been recently studied in [3, 9, 8, 7], for example. To the best of our knowledge, these works differ from each other in the number of features used to characterized fast flux domains and the details of the classification algorithms, but are all limited to mainly studying fast-flux domains advertised through email spam[1]. In particular, given a dataset of spam

[1]The domain names found in domain blacklists and malware samples are also considered in some works, but they are very few compared to the domain names extracted from spam emails.

emails (typically captured by spam traps and filters), potential fast-flux domain names are identified by extracting them from the URLs found in the body of these emails [3, 9, 8, 7]. Then, an *active probing* strategy is applied, which repeatedly issues DNS queries to collect information about the set of resolved IP addresses and to classify each domain name into either *fast-flux* or *non-fast-flux*. The work in [4] is in part different from other previous work, because it is not limited to domains found in spam emails. Hu et al. [4] 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. Also, the work in [4] is heavily based on a DNS analysis module that applies active probing in a way very similar to [3, 9], in order to collect the information necessary to perform the classification of suspicious domains collected from spam emails and the correlation with network flows information.

B. Our Approach

In this paper we propose a novel, *passive* approach for detecting and tracking malicious flux service networks. Our detection system is based on passive analysis of recursive DNS traces collected from multiple large networks. In practice, we deploy a sensor in front of the recursive DNS (RDNS) server of different networks, passively monitor the DNS queries and responses from the users to the RDNS, and selectively store information about potential fast-flux domains into a central DNS data collector. Since the amount of RDNS traffic in large networks is often overwhelming, we devised a number of prefiltering rules that aim at identifying DNS queries to potential fast-flux domain names, while discarding the remaining requests to legitimate domain names. Our prefiltering stage is very conservative, nevertheless, it is able to reduce the volume of the monitored DNS traffic to a tractable amount without discarding information about domain names actually related to malicious flux services. Once information about potential malicious flux domains has been collected for a certain epoch E (e.g., one day), we perform a more fine-grain analysis. First, we apply a clustering process to the domain names collected during E, and we group together domain names that are related to each other. For example we group together domain names that point to the same Internet service, are related to the same CDN, or are part of the same malicious flux network. Once the monitored domain names have been grouped, we classify these *clusters of domains* and the related monitored resolved IP addresses as either being part of a malicious flux service network or not. 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, 9, 7].

Our detection approach has a fundamental advantage, compared to previous work. Passively monitoring *live* users' DNS traffic offers a new vantage point, and allows us to capture 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*, beside email spam and precompiled domain blacklists such as the ones used in [3, 9, 8, 7]. Furthermore, differently from the active probing approach used in previous work [3, 9, 8, 7], we passively monitor live users' traffic without interacting ourselves with the flux networks. Active probing of fast-flux domain names [3, 9, 8, 7] 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.

We implemented a proof-of-concept version of our detection system, and experimented with the RDNS traffic passively collected at two large ISP networks. Overall, our sensors monitored more than 2.5 billion DNS queries per day from milions of distinct source IPs for a period of 45 days. Our experimental results show that the proposed approach is able to accurately distinguish between malicious flux services and legitimate CDNs or other legitimate services. Furthermore, we show how the output of our passive detection and tracking of malicious flux networks may benefit the accuracy of spam filtering applications.

II. Detecting Malicious Flux Networks

In this paper we focus on detecting malicious flux networks *in-the-wild*. We passively observe the RDNS traffic generated by a large user base, and we assume that during their normal Internet experience some of these users will (intentionally or unintentionally) request malicious content served through a flux network. In practice, given the large user base we are able to monitor, it is very likely that at least some of these users will (unfortunately) fall victims of malicious web content, and will therefore "click" on (and initiate DNS

queries about) flux domain names. We aim to detect such events, and track the flux domain names and the IP address of the related flux agents contacted by the victims in the monitored network. Since we perform passive analysis and we monitor real users' activities, this allows us to stealthily detect and collect information about "popular" malicious flux networks on the Internet, regardless of the method used by botmasters to advertise the malicious content served by their flux networks.

A. Characteristics of Flux Domain Names

Fast-flux domains are characterized by the following main features: a) short time-to-live (TTL); b) the set of resolved IPs (i.e., the flux agents) returned at each query changes rapidly, usually after every TTL; 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 [12]. 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 analyzing the value of each single feature may not be enough to precisely identify malicious flux domains and distinguish them from legitimate domains, in Section II-F we will discuss how using a combination of features allows us to accurately separate flux network services from legitimate services.

B. System Overview

Figure 1 presents an overview of our *malicious flux service* detection system. For each RDNS sensor, we monitor the sequence of DNS queries and responses from/to the users' machines for a predefined period of time, or epoch, E (e.g., one day). The amount of DNS traffic towards RDNS servers is often overwhelming, even for medium- and small-size networks. Therefore, our detection system first applies a number of filtering rules to reduce the volume of traffic to be analyzed. Since we are only interested in flux domain names and their resolved IPs, the traffic volume reduction filter F1 is responsible for identifying DNS queries that are most likely related to flux domains, while filtering out queries to domains that are very unlikely to be "fluxing". A list $\mathcal{L}$ of *candidate* flux domain names is kept in memory and updated periodically. This list contains historic information about *candidate* flux domain names, namely the maximum TTL ever seen for each domain name, the set of resolved IPs extracted from the DNS responses over time, etc. At the end of every period $\Delta T < E$ (e.g., ΔT may be equal to a few hours), the list of candidate flux domain names is checked by filter F2 to verify if they are still likely to be flux domains, according to the collected historic information. For example, F2 checks whether the set of resolved IPs returned by the RDNS for a given domain name has grown during ΔT. In fact, if a domain name was queried several times during ΔT, but no new resolved IP was observed, it is unlikely that the domain name is associated to a malicious flux service. On the other hand, if the set of resolved IPs returned by the RDNS for a certain domain name keeps changing after every TTL, the domain name is considered a good candidate flux domain. The domain names that are found not to be likely flux-related are pruned from the list $\mathcal{L}$.

At the end of each epoch E, the remaining candidate flux domains in $\mathcal{L}$ and related historic information are transfered from the RDNS sensors to our *Detector* machine, where we perform further analysis. In particular, in this phase we aim at clustering together domain names related to the same service. We group domains according to their resolved IP sets. Namely, given two candidate flux domain names, if their set of resolved IPs collected during the epoch E intersect (i.e., the two domain names share a significant fraction of resolved IPs), we consider the two domain names as *similar*. Given this notion of IP-based similarity, we apply a hierarchical clustering algorithm to group domain names that are related to each other. In practice, each of the obtained clusters represents a separate *candidate flux service network*. It is worth noting that filter F1 and F2 are very conservative. They will "accept" domains related to malicious flux services, but may also accept a number of domains related to legitimate CDNs, NTP pools, and other legitimate services that share some technical similarities with malicious flux service networks. As a consequence, a cluster of domains may represent a malicious flux service, a legitimate CDN, a pool of NTP servers, etc. Therefore, after clustering each *candidate flux service network* (i.e., each cluster of domain names and the related resolved IPs) is given to a *service classifier*, which is trained to classify each cluster into either *malicious flux service* or *legitimate/non-flux service*. In the following we describe each single component of our detection system more in details.

C. Traffic Volume Reduction (F1)

In order to describe the traffic volume reduction filter F1, we first need to formally define how the

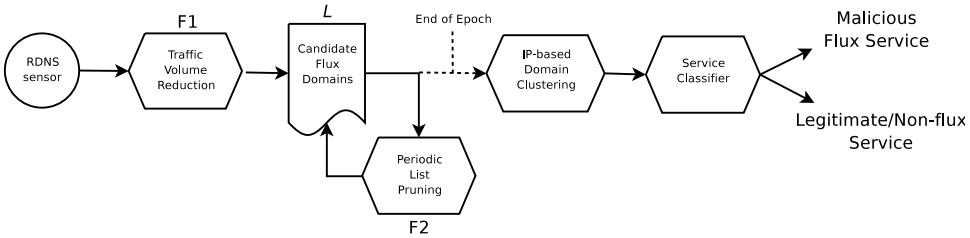

Figure 1: Overview of our detection system.

DNS queries and related responses are represented by our system. Let $q^{(d)}$ be a DNS query performed by a user at time t_i to resolve the set of IP addresses *owned* by domain name d. We formally define the information in the query and its related response as a tuple $q^{(d)} = (t_i, T^{(d)}, \mathbf{P}^{(d)})$, where $T^{(d)}$ is the time-to-live (TTL) of the DNS response, and $\mathbf{P}^{(d)}$ is the set of resolved IPs returned by the RDNS server. Also, let $prefix(\mathbf{P}^{(d)}, 16)$ be the set of distinct `/16` network prefixes extracted from $\mathbf{P}^{(d)}$.

In order to reduce the volume of DNS traffic (see filter F1 in Figure 1) without discarding information about the domain names that are most likely related to malicious flux services, we use the following filtering rules. We "accept" only DNS queries (and related responses) that respect all of the following constraints:

F1-a) $T^{(d)} \leqslant 10800$ seconds (i.e., 3 hours).
F1-b) $|\mathbf{P}^{(d)}| \geqslant 3$ OR $T^{(d)} \leqslant 30$.
F1-c) $p = \frac{|prefix(\mathbf{P}^{(d)},16)|}{|\mathbf{P}^{(d)}|} \geqslant \frac{1}{3}$.

We now briefly motivate the choice of these rules. As mentioned in Section II-A, flux domains are characterized by a low TTL, which is usually in the order of a few minutes [12] and allows the set of resolved IPs to change rapidly. Rule F1-a excludes all the queries to domain names whose TTL exceeds three hours, because such domain names are unlikely to be "fluxing". Rule F1-b takes into account the fact that DNS queries to flux domain names usually return a relatively large number ($\geqslant 3$) of resolved IPs [12]. The reason for this is that the uptime of each flux agent is not easily predictable. A large set of resolved IPs provides a sort of "fault-tolerance" mechanism for the flux service. However, a similar result may also be obtained by setting up flux domains that return a very small set of resolved IPs (e.g., only one per query) but have a very low TTL (e.g., equal or close to zero). This way, if a flux agent is "down", a new flux agent can be immediately discovered by performing another DNS query, because the previous response will be quickly evicted from the RDNS's cache. Rule F1-b takes into account both these scenarios. Rule F1-c is motivated by the fact that the flux agents are often scattered across many different networks and organizations. On the other hand, most legitimate (non-flux) domains resolve to IP addresses residing in one or few different networks. We use the function $prefix(\mathbf{P}^{(d)}, 16)$ to estimate the number of different networks in which the resolved IPs reside[2], and the ratio p (rule F1-c) allows us to identify queries to domains that are very unlikely to be part of a malicious flux service.

D. Periodic List Pruning (F2)

While monitoring the recursive DNS traffic, each sensor maintains a list $\mathcal{L}$ of *candidate* flux domains. The list $\mathcal{L}$ stores historic information about the candidate flux domains and is updated every time a DNS query passes filter F1. In order to explain how $\mathcal{L}$ is updated, let us formally define how a candidate flux domain name is represented. At any time t, a candidate flux domain name d can be viewed as a tuple $d = (t_i, Q_i^{(d)}, \hat{T}_i^{(d)}, \mathbf{R}_i^{(d)}, \mathbf{G}_i^{(d)})$, where t_i is the time when the last DNS query for d was observed, $Q_i^{(d)}$ is the total number of DNS queries related to d ever seen until t_i, $\hat{T}_i^{(d)}$ is the maximum TTL ever observed for d, $\mathbf{R}_i^{(d)}$ is the cumulative set of all the resolved IPs ever seen for d until time t_i, and $\mathbf{G}_i^{(d)}$ is a sequence of pairs $\{(t_j, r_j^{(d)})\}_{j=1..i}$, where $r_j^{(d)} = |\mathbf{R}_j^{(d)}| - |\mathbf{R}_{j-1}^{(d)}|$, i.e., the number of new resolved IPs we observed at time t_j, compared to the set of resolved IPs seen until t_{j-1}. We store only the pairs $\{(t_j, r_j^{(d)})\}_{j=1..i}$ for which $r_j^{(d)} > 0$. Therefore $\mathbf{G}_i^{(d)}$ registers when and how much the resolved IP set of d "grew", until time t_i. When a new DNS query $q^{(d)} = (t_k, T^{(d)}, \mathbf{P}^{(d)})$ related to d passes filter F1, the data structure $d \in \mathcal{L}$ is updated according to the information in $q^{(d)}$.

In order to narrow down the number of candidate flux domains and only consider the ones that are most likely related to malicious flux services, the list $\mathcal{L}$ is pruned at the end of every interval $\Delta T < E$ (e.g., every $\Delta T = 3$ hours). That is, every ΔT we check the status of each candidate flux domain $d \in \mathcal{L}$. Let t_j be the time when this pruning check occurs. Also, let $p = \frac{|prefix(\mathbf{R}_j^{(d)},16)|}{|\mathbf{R}_j^{(d)}|}$

[2]Ideally, it would be better to decide if two IPs belong to two different networks by mapping each IP to their AS number, and then compare the AS numbers. However, it may be hard to do this efficiently, and from our experience using `/16` gives us a way to efficiently compute a good approximation of this result.

be the network prefix ratio (see Section II-C) for the *cumulative* set of resolved IPs of d ever seen until t_j. We remove from $\mathcal{L}$ those domain names for which

F2-a) $Q_j > 100$ AND $|\mathbf{G}_j^{(d)}| < 3$ AND ($|\mathbf{R}_j^{(d)}| \leqslant 5$ OR $p \leqslant 0.5$).

Rule F2-a filters out those domains for which we monitored more than 100 queries, the cumulative set of resolved IPs didn't "grow" more than twice, the total number of resolved IPs ever seen is low ($\leqslant 5$) or the network prefix ratio p is low ($\leqslant 0.5$). The filter F2 is justified by the characteristics of flux domain names described in Section II-A, and domain names that do not pass F2 are very unlikely to be related to flux services.

E. Domain Clustering

At the end of each epoch E, we consider the list $\mathcal{L}$ of candidate flux domains, and we group them 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 service. We speculate that one of the reasons for this behavior is to evade domain blacklists (DBLs).

Our clustering approach groups together domain names that within an epoch E (equal to one day in our experiments) resolved to a common set of IP addresses. To perform domain clustering of flux domains that are related to each other, we use a single-linkage hierarchical clustering algorithm [5, 6], which adopts a "friends of friends" clustering strategy. In order to apply clustering on a set of domain names $\mathcal{D} = \{d_1, d_2, ..d_n\}$, we first need to formally define a notion of similarity between them. Given two domains α and β, and their cumulative set of resolved IP addresses collected during an epoch E, respectively $\mathbf{R}^{(\alpha)}$ and $\mathbf{R}^{(\beta)}$, we compute their similarity score as

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

The first factor is the *Jaccard index* for sets $\mathbf{R}^{(\alpha)}$ and $\mathbf{R}^{(\beta)}$, which intuitively measures the similarity between the two cumulative sets of resolved IPs. The second factor is a sigmoidal weight. In practices, the higher the minimum number of resolved IPs in $\mathbf{R}^{(\alpha)}$ or $\mathbf{R}^{(\beta)}$, the higher the sigmoidal weight. To better understand the choice of this weight factor consider this example: if $|\mathbf{R}^{(\alpha)} \cap \mathbf{R}^{(\beta)}| = 1$ and $|\mathbf{R}^{(\alpha)} \cup \mathbf{R}^{(\beta)}| = 4$ or $|\mathbf{R}^{(\alpha)} \cap \mathbf{R}^{(\beta)}| = 10$ and $|\mathbf{R}^{(\alpha)} \cup \mathbf{R}^{(\beta)}| = 40$, the Jaccard index is 0.25 in both cases. However, in the second case we want the similarity to be higher because there are 10 resolved IPs in common among the domains α and β, instead of just one. We can also think of the second factor as a sort of "confidence" on the first one. The parameter γ is chosen a priori, and is only used to shift the sigmoid towards the right with respect to the x-axes. We set $\gamma = 3$ in our experiments so that if $\min(|\mathbf{R}^{(\alpha)}|, |\mathbf{R}^{(\beta)}|) = 3$ the weight factor will be equal to 0.5. As the minimum number of resolved IPs grows, the sigmoidal weight tends to its asymptotic value of 1.

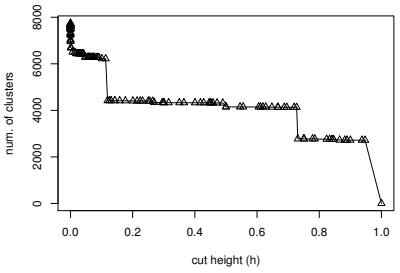

Figure 2: Cluster Analysis, Sensor 1.

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) can then be computed. The hierarchical clustering algorithm takes P as input and produces in output 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 clusters [5]. The obtained *dendrogram* does not actually define a partitioning of the domains into clusters, rather it defines "relationships" among domains. A partitioning of the set $\mathcal{D}$ into clusters can then be obtained by cutting the dendrogram at a certain hight h. In order to choose the best dendrogram cut (i.e., the best clustering), we apply a clustering validation approach based on *plateau* regions [2]. In practice we plot a graph that shows how the number of clusters varies for varying hight of the cut, and we look for *plateau* (i.e., flat) regions in the graph. For example, consider Figure 2, which is related to clusters of candidate flux domain names extracted from an RDNS sensor (see Section III for details) after an epoch $E = 1$ day. The long plateau region between 0.1 and 0.7 shows that varying the cut height h does not significantly change the number of obtained clusters, thus providing for a sort of "natural grouping" of the domain names. A manual validation of the clusters obtained using this analysis strategy confirmed that the obtained clusters were indeed correct. We will discuss the clustering results more in detail in Section III.

F. Service Classifier

Each cluster $\mathcal{C}_i$ of candidate flux domains can be seen as a *candidate flux service* 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. Since filters F1 and F2 adopt a conservative approach, they may not be able to filter out domains related to legitimate CDNs or other legitimate Internet services (e.g., pools of NTP servers) that have a behavior somewhat similar to flux services. Therefore, after collecting and clustering the candidate flux domains we need to determine which clusters are actually related to malicious flux services and which ones are related to *legitimate* and *non-flux* networks. To this end, we apply a statistical *supervised learning* approach to build a network classifier which can automatically distinguish between malicious flux services and other networks, as shown in Figure 1.

We first describe and motivate the set of statistical features we use to distinguish between malicious flux services and legitimate/non-flux services. In [9], Passerini et al. proposed a thorough characterization of fast-flux domain names in terms of statistical features for supervised learning. They introduced a set of nine features based on the analysis of the set of IP addresses resolved by querying single domain names. In this work we adapt some of the features proposed in [9] to characterize clusters of domain names (as opposed to single domains) related to malicious flux services, and we introduce some additional new features. We divided our feature set into two groups, namely "passive" features and "active" features. We call "passive" those features that can be directly extracted from the information collected by passively monitoring the DNS queries from our RDNS sensors. On the other hand, "active" features need some additional *external* information to be computed (e.g., information extracted from `whois` queries, geolocation mapping of IP addresses, BGP announcement data, etc.). For each cluster of domains obtained as described in Section II-E, and related to an epoch E_m, we compute the following features:

"Passive" Features:

- ϕ_1 **Number of resolved IPs**. This is the overall number of distinct resolved IP addresses ever observed during epoch E_m for all the domains in a cluster.
- ϕ_2 **Number of domains**. This is the total number of distinct domain names in a cluster.
- ϕ_3 **Avg. TTL per domain**. Average TTL of the domains in a cluster.
- ϕ_4 **Network prefix diversity** This is the ratio between the number of distinct /16 network prefixes and the total number of IPs. This feature is used to *estimate* the degree of scattering of IP addresses among different networks, and represents a reasonable approximation of "active" features ϕ_7 and ϕ_8 (explained below).
- ϕ_5 **Number of domains per network**. Number of distinct domain names that resolved to at least one of the IP addresses in the considered cluster, during all the previous epochs $E_1, E_2, \ldots$ until the considered epoch E_m. In spite of the high variability of flux domain names, flux networks are rather stable and persistent [7]. Thus, the same flux agents will be used by many distinct domain names, during the time. This feature measures how many domains can be associated to the IPs (i.e., the flux agents) in a cluster, throughout different epochs.
- ϕ_6 **IP Growth Ratio**. This represents the average number of new IP addresses "discovered" per each DNS response related to any domain in a cluster. Namely, $\frac{1}{|\mathcal{C}_i|} \cdot \sum_{d \in \mathcal{C}_i} \frac{|\mathbf{R}^{(d)}|}{Q^{(d)}}$.

"Active" Features:

- ϕ_7 **Autonomous System (AS) diversity**, ϕ_8 **BGP prefix diversity**, ϕ_9 **Organization diversity**. We measure the ratio between the number of distinct ASs where the IPs of a cluster reside and the total number of resolved IPs. Also, we compute similar ratios for distinct organization names and distinct BGP prefixes the IPs in the cluster belong to.
- ϕ_{10} **Country Code diversity**. For each IP in a cluster, we map it to its geographical location and compute the ratio between the number of distinct countries across which the IPs are scattered and the total number of IPs.
- ϕ_{11} **Dynamic IP ratio**. The bot-compromised machines that constitute malicious flux services are mostly *home-user* machines. In order to estimate whether an IP is related to a home-user machine, we perform a reverse (type `PTR`) DNS lookup for each IP, and we look for keyworks such as "dhcp", "dsl", "dial-up", etc., in the DNS response to identify machines that use a dynamic (as opposed to static) IP address. We then compute the ratio between the (estimated) number of dynamic IPs in the cluster and the total number of IPs.
- ϕ_{12} **Average Uptime Index**. This feature is obtained by actively probing each IP in a cluster about six times a day for a predefined number of days (e.g. 5 days), and attempting to establish TCP connections on ports 80/53/443, i.e., HTTP/DNS/HTTPS services[3]. If the host accepts to establish the TCP connection, it is considered *up*, otherwise it is considered *down*. An estimate of the uptime of each IP is given by the ratio between number of times the IP is found to be up versus the total number of probes. Feature ϕ_{12} is computed as the average uptime for the IPs in a cluster.

After measuring the features described above, we employ the popular C4.5 decision-tree classifier [10] to automatically classify a cluster $\mathcal{C}_i$ as either *malicious flux service* or *legitimate/non-flux service*. 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) the decision-tree built during training can be easily interpreted to determine what are the most discriminant features that allow us to distinguish between malicious flux services and legitimate/non-flux services; c) the C4.5 is able to automatically prune the features that are not useful, and potentially create noise instead of increasing classification accuracy [10]. We first train the C4.5 classifier on a *training dataset* containing a number of *labeled* clusters related to malicious flux

[3]We use TCP instead of UDP for DNS probing, because most off-the-shelf DNS software are designed to listen on port 53 for both TCP and UDP communications.

services and clusters related to legitimate/non-flux services. Afterwards, the classifier can be used "online" to classify the clusters obtained at the end of each epoch E from the data collected at each RDNS sensor, as shown in Figure 1. The details of how we obtained the labeled training dataset of malicious flux and legitimate clusters and estimated the accuracy of our service classifier are reported in Section III.

III. Experiments

In this Section we present the results obtained with our malicious flux network detection system. All the experiments related to the clustering of candidate flux domains and classification of flux service networks were conducted on a 4-core 3GHz Intel Xeon Machine with 16GB or memory. However, because the machine was shared with other researchers, we constrained ourselves to using a maximum of 5GB of RAM for our experiments.

A. Collecting Recursive DNS Traffic

We placed two traffic sensors in front of two different RDNS servers of a large north American Internet Service Provider (ISP). These two sensors monitored the RDNS traffic coming from users located in the *north-eastern* and *north-central* United States, respectively. Overall, the sensors monitored the *live* RDNS traffic generated by more than 4 million users for a period of 45 days, between March 1 and April 14, 2009. During this period, we observed an average of about 1.3 billion DNS queries of type `A` and `CNAME` per sensor. Overall we monitored over 2.5 billion DNS queries per day related to hundreds of millions of distinct domain names. The traffic collected at each sensor is reduced using filters F1 and F2, as shown in Figure 1 and described in Section II. We set the epoch E to be one day. The overwhelming traffic volume monitored by the RDNS sensors was effectively reduced from more than 10^9 DNS queries for tens of millions of distinct domain names, to an average of $4 \cdot 10^4$ to $6 \cdot 10^4$ candidate flux domain names per day (depending on the sensor we consider).

B. Clustering Candidate Flux Domains

At the end of each epoch, the candidate flux domains extracted by the RDNS sensors are transfered to our *Detector* machine, where they undergo a clustering process. Before applying domain clustering we further narrow down the number of candidate domain names using a set of filtering rules reported in Appendix. This further filtering step is optional and we mainly use it to reduce the amount of memory required by the clustering algorithm. These filtering rules may be tuned (or eliminated) to "accept" more domains for the clustering step if more memory was available. This additional filter step reduces the average number of candidate flux domains to be clustered of almost an order of magnitude (from $4{\cdot}10^4$ to $6{\cdot}10^4$, to about $8{\cdot}10^3$ domains per sensor). It is worth noting, though, that similarly to filter F1 and F2 (see Section II) the filtering rules reported in Appendix are still very conservative. In fact, from our experimental results we noticed that even after this further filtering, the list of candidate domain names still included all the domain names most likely related to malicious flux services, along with domain names related to legitimate CDNs, pools of NTP servers, and other legitimate services.

Once filtering is completed, we apply a single-linkage hierarchical clustering algorithm [5, 6] to group together domains that belong to the same network, as described in Section II-E. After transfering the data collected from the RDNS sensors to our detection system, the time needed for the clustering process was around 30 to 40 minutes per day and per each sensor. The hight of the dendrogram cut was chosen to be $h = 0.6$. This choice is motivated by the fact that we want to cut the dendrogram at an height within the largest *plateau* region (see Section II-E). In particular, by plotting the cluster analysis graphs similar to the one reported in Figure 2 for different days, we noticed that the value $h = 0.6$ (on the x-axes) was always located around the end of the largest plateau region and provided high quality clusters. Using $h = 0.6$ we obtained an average of about 4,000 domain clusters per day.

Clustering is a completely unsupervised process [5, 6], and automatically verifying the results is usually very hard if at all possible. Therefore, with the help of a graphical interface that we developed, we manually verified the quality of the results for a subset of the clusters obtained every day. In particular, in order to assess the quality of the domain clusters, we manually verified that the domain names in a cluster were actually related to the same "service" (e.g., the same CDN, the same malicious flux network, the same NTP pool, etc.). In many cases this manual evaluation was straightforward. For example, our clustering algorithm was able to correctly identify clusters of domain names belonging to a malicious flux service that was being used for phishing facebook login credentials. In this case the domain names advertised by the botmaster all shared very strong structural similarities because they all started with "login.facebook", contained a string of the form "personalid-RAND", where "RAND" is a pseudo

random string, and ended with ".com". Also, our IP-based clustering process (see Section II-E) was able to correctly group together domain names related to the NTP server pool in Europe and separate them from the group of domains related to the NTP pool in North America, the pool of domains related to Oceania, etc. The domain names related to the NTP pools in different regions of the world can visually be distinguished from each other. Therefore, it was easy to verify that domains such as 0.europe.pool.ntp.org, uk.pool.ntp.org, fr.pool.ntp.org were all correctly grouped together, and separated from the cluster containing au.pool.ntp.org and oceania.pool.ntp.org, for example. In other cases we had to confirm the correctness of our clusters by manually probing the clustered domain names and finding relations between the obtained resolved IPs and the services (e.g., web pages) provided through them.

C. Evaluation of the Service Classifier

In this section we explain the results related to the classification of clusters of domains into either *malicious flux services* or *legitimate/non-flux services*. As described in Section II-F, we use a statistical *supervised learning* approach to build a *service classifier*. In order to use a supervised learning approach, we first need to generate a dataset of labeled clusters (the "ground-truth") which can be used to train our statistical classifier, and evaluate its classification accuracy. We first describe how this *labeled dataset* was generated, and then motivate why the different statistical features used by the classifier, and described in detail in Section II-F, allow us to accurately detect malicious flux service networks.

In order to construct the labeled dataset for our experiments, we manually inspected and labeled a fairly large number of clusters of domains generated by the clustering process described in Section II-E. To make the labeling process less time consuming, we developed a graphical interface that allows us to rank clusters of domains according to different features. For example, our interface allows us to rank all the clusters according to their *network prefix* diversity (feature $\phi_{\mathbf{4}}$), the cumulative number of distinct resolved IPs (feature $\phi_{\mathbf{1}}$), the IP growth ratio (feature $\phi_{\mathbf{6}}$), etc. In addition, our graphical interface allows us to inspect several other properties of the clusters, such as CNAME entries collected from DNS responses, the content of Web pages obtained by contacting a sample of resolved IPs from a cluster, information gathered from queries to `whois` and search engines, etc.

As we discussed in Section II-F, we use the C4.5 decision tree classifier to automatically classify between clusters of domains related to malicious flux networks and clusters related to legitimate or non-flux networks. We discuss the details of how we trained and tested our classifier later in this section. Here it is important to notice that one of the reasons we chose the C4.5 classifier is that the decision tree obtained after the training period is relatively easy to interpret [10]. In particular, we noticed that when using the "passive" features described in Section II-F for training, the classifier indicated that the *IP Growth Ratio* (feature $\phi_{\mathbf{6}}$) is the most discriminant feature (i.e., the *root* of the decision tree). This confirms the fact that the rapid change of the resolved IPs of flux domains is a distinctive characteristic of malicious flux service networks.

	AUC	DR	FP
All Features	0.992 (0.003)	99.7% (0.36)	0.3% (0.36)
Passive Features	0.993 (0.005)	99.4% (0.53)	0.6% (0.53)
$\phi_{\mathbf{6}}$, $\phi_{\mathbf{3}}$, $\phi_{\mathbf{5}}$	0.989 (0.006)	99.3% (0.49)	0.7% (0.49)

Table I: Classification performance computed using 5-fold cross-validation. AUC=Area Under the ROC Curve; DR=Detection Rate; FP=False Positive Rate. The numbers between parenthesis represent the standard deviation of each measure.

Since we focus on classifying malicious flux services, and considering that the number of flux agents for each flux service network is usually very high, we only consider clusters of domains for which overall we observed at least $\phi_{\mathbf{1}} \geqslant 10$ resolved IPs. With the help of our graphical interface, during the entire month of March 2009 we were able to label 670 clusters as being related to malicious flux networks of various kinds (e.g., flux networks serving malware, adult content, phishing websites, etc.), and 8,541 clusters related to non-flux/legitimate clusters, including clusters related to different CDNs, NTP pools, IRC pools, and other legitimate services. Using this labeled dataset and a 5-fold cross-validation approach, we evaluated the accuracy of our classifier. The obtained results are reported in Table I. It is easy to see that our network classifier is able to reach a very high Area Under the ROC curve [1], and a high detection rate and low false positive rate at the same time. We performed experiments using three different sets of features. First, we used all the "passive" and "active" features described in Section II-F to characterize clusters of domains. Afterwards, we repeated the same experiments using only the "passive" features (second row in Table I). From this last experiments, the C4.5 learning algorithm generated a decision tree whose root was feature $\phi_{\mathbf{6}}$, and with feature $\phi_{\mathbf{3}}$ and $\phi_{\mathbf{5}}$ as children nodes at the top of the tree. This indicates that these three features tend to be the most "useful" ones for distinguishing between malicious flux networks and legitimate networks. For the sake of comparison, we

evaluated the classification performance of our classifier using only these three features. As we can see from the third row in Table I, only three features are sufficient to obtain very good classification results, although using all the available features produces slightly better results. Furthermore, we evaluated our classifier in an operational setting. Namely, we used the entire labeled dataset described above to train our network classifier, and then we used the obtained classifier to classify new (i.e., not seen during training) clusters of domains obtained in the first 14 days of April. During this period we obtained an average of 448 clusters per day (again, considering only clusters for which $\phi_{\mathbf{1}} \geqslant 10$), 26 of which (in average per day) where classified as being related to flux service networks. We manually verified that the classifier correctly labeled clusters related to malicious flux networks with very few false positives, thus confirming the results reported in Table I. Overall, during the entire 45 days period of evaluation, between March 1 and April 14, 2009, we detected an average of 23 malicious flux service networks per day, with a total of 61,710 flux domain names and 17,332 distinct IP addresses related to flux agents.

D. Can this Contribute to Spam Filtering?

In this Section we analyze to what extent the information about malicious flux networks passively gathered at the RDNS level using our detection system can benefit spam filtering applications. In particular, we focus on detecting whether domain names found in spam emails are somehow related to malicious flux networks detected by our system. Assume a mail server receives an email which contains a link to a certain website, performs a DNS query for the domain name embedded in the link, and forwards the email to the spam filter along with the obtained set, say $\mathbf{R}_f$, of resolved IP addresses. At this point the spam filter can inspect the content of the email, and also check if there is any intersection between the set $\mathbf{R}_f$ and any of the malicious flux networks identified by our detection system. If a significant intersection (i.e., common IP addresses) is found, the spam filter can increase the spam score related to the email, and use this information for making a more accurate overall decision about whether the received email should be classified as *spam* or not. The intuition is that if the domain name of the website advertised via email points to one or more previously detected flux agents, it is very likely that the content of the advertised website is malicious. A similar spam detection process may be used for types of spam different from email spam. For example, using a browser plug-in this spam detection process may be extended to identify *blog spam*, *social network spam*, etc., before the users access the malicious content. Therefore, it is easy to see that the output of our malicious flux detection system may contribute to a number of different spam filtering applications.

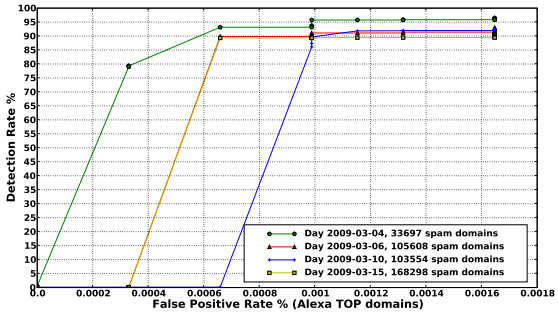

Figure 3: Detection of domains in spam emails.

In order to measure to what extent our detection system may benefit email spam filtering, we proceeded as follows. We obtained a feed of URLs extracted from spam emails captured by a mid-size email *spam-trap* during a period of 30 days, between March 1 and March 30, 2009. This feed provided us with an average of 250,000 spam-related URLs per day, from which we extracted about 86,000 new (i.e., never seen before) domain names per day. Let D_{k+1} be the set of spam-related domain names collected on day $k+1$, $\mathbf{S}^{(d)}_{k+1}$ be the set of resolved IPs obtained by performing one DNS query for domain $d \in D_{k+1}$, and $\hat{\mathbf{R}}^{k}_{k-l}$ be the overall set of IP addresses of flux agents detected by our malicious flux network detection system in the period from day $k-l$ to day k. In order to obtain a *suspiciousness score* $s(d)$ for domain d, we use the similarity metric defined in Equation 1 to compute $s(d) = sim(\mathbf{S}^{(d)}_{k+1}, \hat{\mathbf{R}}^{k}_{k-l})$, i.e., the degree of overlap between the resolved IPs of domain d and the malicious flux networks we were able to detect from RDNS traffic. If the value $s(d)$ exceeds a predefined threshold θ, we classify the domain name d as *malicious*, otherwise we classify it as *legitimate*. We repeat this process for each spam-related domain name $d \in D_{k+1}$ to estimate the detection rate of the proposed approach, namely the percentage of spam-related domain names that may be identified by a spam filter. Furthermore, we considered the list of domain names related to the top 50,000 most popular web sites according to Alexa (www.alexa.com) to estimate the false positives that may be generated by our detection approach. Let A be the set of these popular websites. For each domain $\alpha \in A$, we perform a DNS query and collect the set resolved IP addresses $\mathbf{R}^{(\alpha)}$, and we compute the similarity score $s(\alpha) = sim(\mathbf{R}^{(\alpha)}, \hat{\mathbf{R}}^{k}_{k-l})$. Again, if $s(\alpha) > \theta$ we classify α as malicious. In our experiments, we assume the domain names in the

set A are legitimate/non-flux domains. Therefore, any domain α for which $s(\alpha) > \theta$ is considered as a *false positive*. Figure 3 reports the ROC curves (i.e., the trade-off between false positive and detection rate) obtained by varying the detection threshold θ, using a fixed value of $l = 2$, and for four different value of k (i.e., four different days). It is easy to see that the detection approach described above produces a detection rate of domain names advertised through spam emails between 90% to 95%. It is worth noting that not all of the detected malicious domains are necessarily fast-flux domains. We noted that several of the domain names detected as malicious did not appear to have a "fluxing" behavior themselves, but resolved to a fixed set of IP addresses that partially intersected with the IP addresses of flux agents we detected from our passive analysis of RDNS traffic. A more detailed analysis revealed that these fixed sets of flux agents consisted of machines with a high average uptime. Therefore, we speculate that highly reliable compromised machines may be used as part of larger flux service networks, as well as "stand-alone" providers of malicious content. It is also worth noting that the false positive rate of our approach for detecting spam-related malicious domains is less than 0.002%. This confirms that our malicious flux network detection system is a very promising approach and may substantially benefit spam filtering applications.

IV. Conclusion

In this paper we presented a novel, passive approach for detecting malicious flux service networks *in-the-wild*. Our detection system is based on passive analysis of recursive DNS (RDNS) traffic traces. Contrary to previous work, our approach is not limited to the analysis of suspicious domain names extracted from spam emails or precompiled domain blacklists. Instead, we are able to detect malicious flux service networks as they are accessed by users who fall victims of malicious content advertised through different forms of spam. We experimented with the RDNS traffic passively collected at two large ISP networks, and showed that the proposed approach is able to accurately detect malicious flux service networks. Furthermore, we showed how our passive detection and tracking of malicious flux service networks may benefit spam filtering applications.

Acknowledgments

We would like to thank Giorgio Giacinto and the anonymous reviewers for their helpful comments on earlier versions of this paper.

This material is based upon work supported in part by the National Science Foundation under grants no. 0716570 and 0831300, the Department of Homeland Security under contract no. FA8750-08-2-0141, the Office of Naval Research under grant no. N000140911042. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the Department of Homeland Security, or the Office of Naval Research.

References

[1] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[2] R. Dugad and N. Ahuja. Unsupervised multidimensional hierarchical clustering. In *International Conference on Acoustics, Speech and Signal Processing*, 1998.

[3] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and detecting fast-flux service networks. In *Network & Distributed System Security Symposium*, 2008.

[4] X. Hu, M. Knysz, and K. G. Shin. Rb-seeker: Auto-detection of redirection botnets. In *Network & Distributed System Security Symposium*, 2009.

[5] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

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

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

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

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[11] SSAC. SAC 025 - SSAC advisory on fast flux hosting and DNS, 2008.

[12] The Honeynet Project. Know your enemy: Fast-flux service networks, 2007.

Appendix

In the additional filtering step F3, we apply the following filtering rules (see Section II-D for the notation), where the subscript E indicates that the quantities are measured at the end of an epoch $E = 1$ day:

F3-a) $T_E < 30$ F3-b) $|\mathbf{R}_E^{(d)}| \geqslant 10$
F3-c) $|\mathbf{G}_E^{(d)}| \geqslant 5$ F3-d) $|\mathbf{R}_E^{(d)}| \geqslant 5$ AND $p_E \geqslant 0.8$
F3-e) $p_E \geqslant 0.5$ AND $T_E \leqslant 3600$ AND $|\mathbf{G}_E^{(d)}| \geqslant 10$