Phoenix: DGA-Based Botnet Tracking and Intelligence*

Stefano Schiavoni[1], Federico Maggi[1], Lorenzo Cavallaro[2], and Stefano Zanero[1]

[1] Politecnico di Milano, Italy
[2] Royal Holloway, University of London, UK

Abstract. Modern botnets rely on domain-generation algorithms (DGAs) to build resilient command-and-control infrastructures. Given the prevalence of this mechanism, recent work has focused on the analysis of DNS traffic to recognize botnets based on their DGAs. While previous work has concentrated on detection, we focus on supporting intelligence operations. We propose Phoenix, a mechanism that, in addition to telling DGA- and non-DGA-generated domains apart using a combination of string and IP-based features, characterizes the DGAs behind them, and, most importantly, finds groups of DGA-generated domains that are representative of the respective botnets. As a result, Phoenix can associate previously unknown DGA-generated domains to these groups, and produce novel knowledge about the evolving behavior of each tracked botnet. We evaluated Phoenix on 1,153,516 domains, including DGA-generated domains from modern, well-known botnets: without supervision, it correctly distinguished DGA- vs. non-DGA-generated domains in 94.8 percent of the cases, characterized families of domains that belonged to distinct DGAs, and helped researchers "on the field" in gathering intelligence on suspicious domains to identify the correct botnet.

1 Introduction

The malware-as-a-service trend is resulting in an increasing number of small, distinct botnets, which are predicted to replace larger ones [11]. Because of their size, they can fly under the radar of malware analysts. Keeping track of such a diverse population and traffic patterns is difficult. The typical objective of botnet intelligence is to find the addresses or domain names of the command-and-control (C&C) server of a botnet, with the goal of *sinkholing* it.

Albeit some botnets use P2P protocols to remove single points of failure, domain-generation algorithms (DGAs) are still in wide use. As detailed in §2 and 7, researchers have proposed various approaches for finding and characterizing *individual* DGA-generated domains. However, such approaches require visibility of the original DNS queries, complete with source IP addresses. This requires

* This research has been funded by EPSRC G.A. EP/K033344/1 and EU FP7 n.257007. The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

S. Dietrich (Ed.): DIMVA 2014, LNCS 8550, pp. 192–211, 2014.
© Springer International Publishing Switzerland 2014

low-level DNS sensors to be deployed between the infected machines and their DNS servers. This entails privacy issues and restricts operation of such schemes to network administrators of large networks. In addition, the accuracy of client-IP-based approaches is affected by IP-sharing mechanisms (e.g., NAT).

A higher-level observation point is beneficial both in terms of ease of deployment and of scope. We propose Phoenix, which requires only publicly available DNS traffic and an initial feed of malicious domains (not necessarily DGA-generated). With this information, we (1) find DGA-generated domains, (2) characterize the generation algorithms, (3) isolate logical groups of domains that represent the respective botnets, and (4) produce novel knowledge about the evolving behavior of each tracked botnet. Phoenix requires no prior knowledge of the DGAs nor reverse engineering of malware samples. Being based on recursive-level DNS traffic, our approach guarantees repeatability [16] and preserves the privacy of the infected computers, by not requiring any data about them.

In brief, Phoenix first models *pronounceable* domains, likely to be generated by a human user, and considers DGA-generated those which violate the models (thus, not making use or learning the characteristics of specific DGAs). In particular, we apply such filter to well-known blacklists of malicious domains, finding those that are likely to be DGA-generated as well as malicious. Our technique is unsupervised, and allows to set the amount of acceptable error a priori (see § 4.1). Phoenix then groups these domains according to the domain-to-IP relations. This step also filters out DGA-looking domains that are benign (e.g., a benign acronym which happens to be unpronounceable). Phoenix then derives a generic set of fingerprints useful to label new malicious DGA domains, track botnets' evolution, or gather insights on their activity (e.g., C&C migrations).

Notably, on Feb 9th, 2013 we obtained an undisclosed list of DGA-generated domains for which no knowledge of the respective botnet was available before. Phoenix correctly labeled these unknown domains as belonging to Conficker.

2 Background and Research Gaps

While botnets with a fully P2P topology are on the rise, DNS is still abused by cybercriminals to build *centralized, yet reliable* botnet infrastructures [2, 3, 8, 14, 15, 21]. An effective technique used to improve resiliency to take downs and tracking is domain flux. In such botnets, the bots and the C&C servers implement the same algorithm to generate a large and time-dependent list of domain names based on pseudo-unpredictable seeds. Only one of these DGA-generated domains is actually registered and pointing to the true IP address of the C&C. The bots will then generate and query all these domains, according to the DGA, until a DNS server answers with a non-NXDOMAIN reply, that is the IP address of the respective (existing) domain. Only the DGA authors know exactly when the upcoming rendezvous domain has to be registered and activated, and this avoids the shortcomings that in past allowed researchers to take over botnets [19].

DGA-based botnets are still prevalent (see, e.g., https://blog.damballa.com/archives/1906, or http://threatpost.com/pushdo-malware-resurfaces-with-dga-capabilities). Finding groups of related DGA-generated domains provides valuable insights to recognize bots that belong to the same botnet, or to a set of botnets that share a similar DGA. With this knowledge, analysts can follow their evolution and their (changing) C&Cs over time, where these are hosted, and the number of machines involved. The task of finding families of related DGA-generated domains, however, is tedious and labor-intensive, although previous research has devised mechanisms to partially automate it. Reverse-engineering a DGA still requires effort and, in most of the cases, a malware sample. In this work, we show how instances of domains names generated from the same DGA can be generalized to "fingerprint" the generation algorithm itself.

A side effect of DGA mechanisms is that each infected machine performs a large amount of DNS queries that yield NXDOMAIN replies. Legitimate hosts have no reasons to generate high volumes of such queries. This observation has been leveraged by Antonakakis et al. [3] to detect DGA-based bots. Unfortunately, as also noticed by Perdisci et al. [15], this criterion requires to know the IP addresses of the querying hosts. An alternative technique is proposed in [20], who grouped together DNS queries originated by the same client to define the correlation between distinct requests that target the same domains.

These approaches are very interesting to detect infected clients over a large network over which the analyst has full control. However, they impose undesirable requirements in terms of input data and deployment to create a large-scale observatory and intelligence service. First, relying on the IP addresses of querying hosts is error prone, because of IP-(re)assignment and masquerading policies employed by ASs. More importantly, access to this information is limited in scope, because it is available only from DNS servers placed *below* the recursive DNS level (e.g., host DNSs). This can be problematic for researchers, but also for practitioners who want to operate these systems beyond the scope of a single network. Finally, of particular interest for researchers, IP information raises privacy concerns, leading to non-repeatable experiments [16], as datasets that include these details cannot be made publicly available.

Modeling and characterizing a DGA from the sole domain name is indeed hard, in particular when observing one domain at a time, because one sample is not representative of the whole random generation process. Grouping domain samples to extract the characteristics of the DGA is also challenging: How to group domains together, or avoid spurious samples that would bias the results?

3 System Overview

Phoenix is divided into three modules, as shown in Fig. 1. The core **Discovery** module identifies and models DGA-generated domains. The **Detection** module receives one or more domain names with the corresponding DNS traffic, and uses the models built by the **Discovery** module to tell whether such domain names appear to be automatically generated. If that is the case, this module

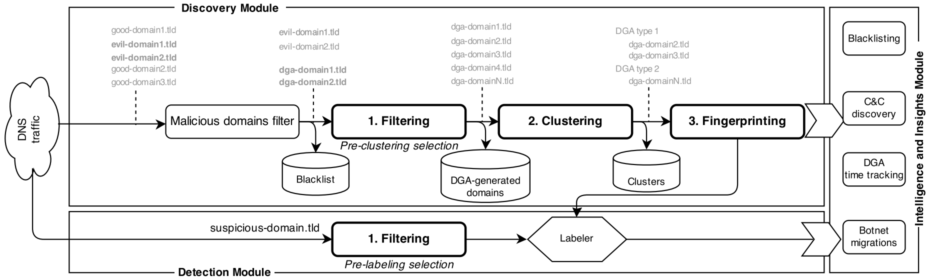


Fig. 1. The **Discovery** module processes the domain names from a domain reputation system and identifies DGA-generated domains. The **Detection** module analyzes a stream of DNS traffic and recognizes the (previously unknown) domains that resemble a known DGA. The **Intelligence and Insights** module provides the analyst with information useful, for instance, to track a botnet.

labels those domains with an indication of the DGA that is most likely behind the domain generation process. Last, the **Intelligence and Insights** module aggregates, correlates and monitors the results of the previous modules to extract meaningful information from the observed data (e.g., whether an unknown DGA-based botnet is migrating across ASs).

3.1 Discovery Module

This module discovers domains that exhibit DGA characteristics. It receives two input streams. One is a stream of domain names that are generically known to be malicious. Any blacklist or domain reputation system (e.g., Exposure [6]) can be used as a source. The second input is a stream of DNS queries and replies related to such domains and collected above the recursive resolvers, for instance by a passive and privacy-preserving DNS monitor (e.g., SIE). The blacklists that we rely on are generated from privacy-preserving DNS traffic too.

Step 1 (Filtering). We extract a set of *linguistic features* from the domain names. The goal is to recognize the ones that appear to be the results of automatic generation. For ease of explanation and implementation, Phoenix considers the linguistic features based on the English language, as discussed in §6.

Differently from previous work, we devised our features to work well on single domains. Antonakakis et al. [3], Yadav et al. [21, 22], instead, relied on features extracted from *groups* of domains, which creates the additional problem of how to create such groups. The authors circumvented this problem by choosing *random* groups of domains. However, there is no rigorous way to verify the validity of such assumptions. Therefore, as part of our contributions, we made an effort to design features that require no groupings of domains. We make no assumptions about the type of DGA that have generated the domains, although we do assume that at least one exists.

The output is a set of domains, possibly generated by different DGAs. As opposed to requiring DGA-generated domains for training, we use a *semi-supervised technique* which requires limited knowledge on benign, non-DGA-generated domains. The rationale is that obtaining a dataset of these domains is straightforward and not lto a specific DGA. At runtime, in case **Step 1** lets some benign, DGA-looking domains through (e.g., `<ZIP>.com`), **Step 2** will remove them.

Step 2 (Clustering). We extract *IP-based features* from the DNS traffic of the domains that have passed **Step 1**. We use these features to *cluster* together the domains that have resolved to similar sets of IP addresses—possibly, the C&C servers. For example, if `5ybdiv.cn` and `hy093.cn` resolved to the same pool of IPs, we cluster them together. Here, we assume that domains generated by different DGAs are used by distinct botnets/variants, or at least by different botmasters, who have crafted a DGA for their C&C strategy. Therefore, this partitioning to some extent mirrors the different groups of botnets.

Step 3 (Fingerprinting). We extract *other* features from the clusters to create models that define the fingerprints of the respective DGAs. The **Detection** module uses these fingerprints as a lookup index to identify the DGA to which domains never seen before belong. For instance, `epu.org` and `xmsyt.cn` will match two distinct fingerprints. The notion of similarity is by no means based solely on linguistic similarity: We do consider other IP- and DNS-based features. The output is a set of clusters with their fingerprints.

3.2 Detection Module

This module receives in input a (previously unseen) domain name d, which can be either malicious or benign, and uses once again the **Filtering** step to verify whether it is automatically generated. Domain names that pass this filter undergo further checks, which may eventually flag them as not belonging to any cluster (i.e., not matching any of the fingerprints). Therefore, in this step, flagging as “DGA generated” a (benign) domain that does not belong to some DGA is not a major error. It is instead more important not to discard suspicious domains, in order to maximize the recall. Therefore, for this module only, we configure the **Filtering** step with looser parameters (as described in §4.1), so that we do not discard any domains that may be automatically generated. Then, this module leverages the cluster fingerprints to characterize the DGA, if any, that lies behind the previously unseen domain, d.

3.3 Intelligence and Insights Module

The outcome of previous modules builds novel knowledge, by creating clusters of related domains, by fingerprinting their underlying DGA, and by associating new domains to such clusters. With this knowledge, the addresses of the C&C servers and lists of DGA-generated domains can be easily grouped together and associated. With this information, analysts can track separately the evolution of the IPs that the groups point to, and use this information to take action.

For example, recognizing when a C&C is migrated to a new AS is easier when the set of IPs and domains is small and the characteristics of the DGA are known and uniform.

Generally speaking, these analyses can lead to high-level intelligence observations and conjectures, useful for the mitigation of DGA-related threats, for which we provided two use cases in §5.4. In this, we advance the state of the art by providing a tool that goes beyond blacklists and domain reputation systems.

4 System Details

We implemented PHOENIX in Python using the NumPy package, for statistical functions, and the SciPy [9] package, for handling sparse matrices. The deployment is as easy as running a script for each module (§5).

Notation (Domain Names and Suffixes). For the purpose of this work, a *domain name* is a sequence of labels separated by dots (e.g., `www.example.com`) containing a *chosen prefix* (e.g., `example`) and a *public suffix* (e.g., `.com`, `.co.uk`). The public suffix, or top-level domain (TLD), can contain more than one label (e.g., `.co.uk`). The term effective TLD (eTDL) is thus more correct. A domain name can be organized hierarchically into more subdomains (e.g., `www.example.com`, `ssh.example.com`). We only consider the first level of a chosen prefix: A DGA that works on further levels makes little sense, as the first level would still be the single point of failure. Unless clear from the context, we use the terms *domain*, *chosen prefix*, or *prefix* as synonyms.

4.1 Step 1: Filtering

We assume that domains generated by DGAs exhibit different linguistic features than domains crafted by humans with benign intentions. Except for the corner cases discussed in §6, this assumption is reasonable because benign domains have the primary purpose of being easily remembered and used by human beings, thus are usually chosen to meet this goal. On the other hand, DGA-generated domains exhibit a certain degree of linguistic randomness, as numerous samples of the same randomized algorithm exist.

Linguistic Features. Given a domain d and its prefix $p = p_d$, we extract two classes of linguistic features to build a 4-element feature vector for each d. Pilot experiments showed that using multiple features avoids mistakes due to, for instance, artificial brand names.

LF1*: Meaningful Characters Ratio.* Models the ratio of characters of the string p that comprise a meaningful word. Low values indicate automatic algorithms. Specifically, we split p into n meaningful subwords w_i of at least 3 symbols: $|w_i| \geq 3$, leaving out as few symbols as possible: $R(d) = R(p) = \max(\sum_{i=1}^{n} |w_i|)/|p|$. If

$p = \texttt{facebook}$, $R(p) = (|\texttt{face}| + |\texttt{book}|)/8 = 1$, the prefix is fully composed of meaningful words, whereas $p = \texttt{pub03str}$, $R(p) = (|\texttt{pub}|)/8 = 0.375$.

LF2: *n-gram Normality Score.* This class of features captures the pronounceability of a domain name. This is a well-studied problem in linguistics, and can be reduced to quantifying the extent to which a string adheres to the phonotactics of the (English) language. The more permissible the combinations of phonemes [4, 18], the more pronounceable a word is. Domains with a low number of such combinations are likely DGA-generated. We calculate this class of features by extracting the n-grams of p, which are the substrings of p of length $n \in \{1, 2, 3\}$, and counting their occurrences in the (English) language dictionary[1]. If needed, the dictionary can be extended to include known benign, yet DGA-looking names. The features are thus parametric to n: $S_n(d) = S_n(p) := (\sum_{n\text{-gram } t \text{ in } p} \text{count}(t))/(|p| - n + 1)$, where count$(t)$ are the occurrences of the n-gram t in the dictionary. For example, $S_2(\texttt{facebook}) = \texttt{fa}_{109} + \texttt{ac}_{343} + \texttt{ce}_{438} + \texttt{eb}_{29} + \texttt{bo}_{118} + \texttt{oo}_{114} + \texttt{ok}_{45} = 170.8$ seems a non-automatically generated, whereas $S_2(\texttt{aawrqv}) = \texttt{aa}_4 + \texttt{aw}_{45} + \texttt{wr}_{17} + \texttt{rq}_0 + \texttt{qv}_0 = 13.2$ seems automatically generated.

Statistical Linguistic Filter. Phoenix uses **LF1-2** to build a feature vector $\boldsymbol{f}(d) = [R(d), S_{1,2,3}(d)]^T$. It extracts these features from a dataset of benign, non-DGA-generated domains (Alexa top 100,000) and calculates their mean $\boldsymbol{\mu} = \left[\overline{R}, \overline{S_1}, \overline{S_2}, \overline{S_3}\right]^T$ and covariance (matrix) $\boldsymbol{C}$, which respectively represent the statistical average values of the features and their correlation. Strictly speaking, the mean defines the centroid of the dataset in the features' space, whereas the covariance identifies the shape of the hyperellipsoid around the centroid containing all the samples. Our filter constructs a confidence interval, with the shape of such hyperellipsoid, that allows us to separate non-DGA- from DGA-generated domains with a measurable, statistical error that we can set a priori.

Distance Measurement. To tell whether a previously unseen domain d' resembles the typical features of a non-DGA-generated domain, the filter measures the distance between the feature vector $\boldsymbol{f}(d') = \boldsymbol{x}$ and the centroid. To this end, we leverage the Mahalanobis distance: $\text{d}_{Mah}(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$. This distance has the property of (1) taking into account the correlation between features—which is significant, because of how the features are defined, and (2) operating with scale-invariant datasets.

Distance Threshold. A previously unseen domain d' is considered as DGA-generated when its feature vector identifies a point that is too distant from the centroid: $\text{d}_{Mah}(\boldsymbol{x}) > t$. To take a proper decision we define the threshold t as the p-percentile of the distribution of $\text{d}_{Mah}(\boldsymbol{x})$, where $(1 - p)$ is the fraction of non-DGA-generated domains that we allow to confuse as DGA-generated domains. In this way, we can set the error a priori. As mentioned in §3.2, the **Discovery** module employs a strict threshold, $t = \Lambda$, whereas the **Detection** module requires a looser threshold, $t = \lambda$, where $\lambda < \Lambda$.

Threshold Estimation. To estimate proper values for λ and Λ, we compute $\text{d}_{Mah}(\boldsymbol{x})$ for $\boldsymbol{x} = \boldsymbol{f}(d), \forall d \in \mathbb{D}_{HGD}$, whose distribution is plotted in Fig. 2a

[1] In our implementation we used `http://tinyurl.com/top10000en`

as ECDF. We then set Λ to the 90-percentile and λ to the 70-percentile of that distribution, as annotated in the figure. Fig. 2b depicts the 99%-variance preserving 2D projection of the hyperellipsoid associated to $\mathbb{D}_{HGD}$, together with the confidence interval thresholds calculated as mentioned above.

4.2 Step 2: Clustering

This step receives as input the set of domains $d \in \mathbb{D}$ that have passed **Step 1**. These domains are such that $\mathrm{d}_{Mah}(\boldsymbol{f}(d)) > \Lambda$, which means that d is likely to be DGA-generated, because they are too far from the centroid.

The goal of this step is to cluster domains according to their similarity. We define as *similar* two domains that resolved to "similar" sets of IP addresses. The rationale is that the botmaster of a DGA-based botnet registers several domains that, at different points in time, resolve to the same set of IPs (i.e., the C&C servers). To find similar domains, we represent the domain-to-IP relation as a bipartite graph, which we convert in a proper data structure that allows us to apply a spectral clustering algorithm [13]. This returns the groups of similar domains (i.e., nodes of the graph). In this graph, two sets of node exists: $K = |\mathbb{D}|$ nodes represent the domains, and $L = |\,\mathrm{IPs}(\mathbb{D})|$ nodes represent the IPs. An edge exists from node $d \in \mathbb{D}$ to node $l \in \mathrm{IPs}(\mathbb{D})$ whenever a domain pointed to an IP.

Bipartite Graph Recursive Clustering. To cluster the domain nodes $\mathbb{D}$, we leverage the DBSCAN clustering algorithm [7].

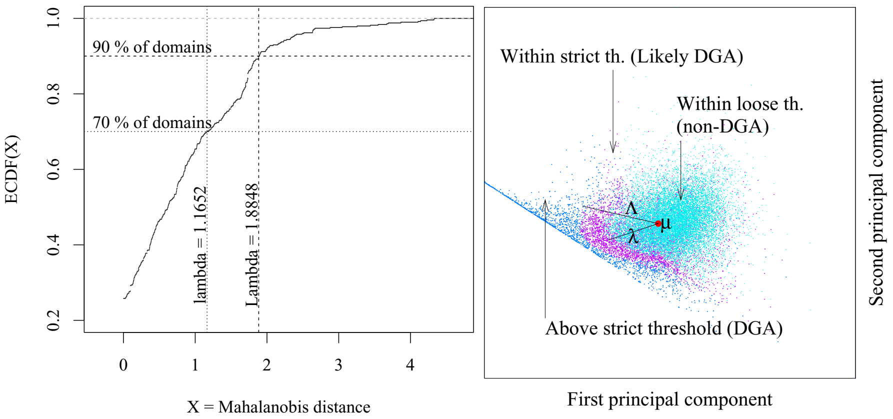
Data Structure. We encode the bipartite graph as a sparse matrix $\boldsymbol{M} \in \mathbb{R}^{L \times K}$ with L rows and K columns. Each cell $M_{l,k}$ holds the weight of an edge $k \rightarrow l$ in the bipartite graph, which represents the fact that domain d_k resolves to IP l. The weight encodes the "importance" of this relation. For each IP l in the graph, the weights $M_{l,k}, \forall k = 1, \ldots, K$ are set to $\frac{1}{|\mathbb{D}(l)|}$, where $\mathbb{D}(l) \subset \mathbb{D}$ is the subset of domains that point to that IP. This weight encodes the peculiarity of each IP: The less domains an IP is pointed by, the more characterizing it is.

Domain Similarity. We calculate the matrix $\boldsymbol{S} \in \mathbb{R}^{K \times K}$, whose cells encode the similarity between each pair of domains d and d'. We want to consider two domains as highly similar when they have peculiar IPs in common. Therefore, we calculate the similarity matrix from the weights, as $\boldsymbol{S} = \boldsymbol{N}^T \cdot \boldsymbol{N} \in \mathbb{R}^{K \times K}$, where $\boldsymbol{N}$ is basically $\boldsymbol{M}$ normalized by columns (i.e., $\sum_{l=1}^{L} M_{l,k} = 1, \forall k = 1, K$). This similarity matrix implements the rationale that we mentioned at the beginning of this section.

Domain Features and Clustering. We apply the DBSCAN algorithm hierarchically. We compute the first normalized eigenvector $\boldsymbol{v}$ from $\boldsymbol{S}$. At this point, each domain name d_k can be represented by its feature v_k, the k-th element of $\boldsymbol{v}$, which is fed to the DBSCAN algorithm to produce the set of R clusters $\mathcal{D} = \{\mathbb{D}^1, \ldots, \mathbb{D}^R\}$ at the current recursive step.

Clustering Stop Criterion. We recursively repeat the clustering process on the newly created clusters until one of the following conditions is verified:

- a cluster of domains $\mathbb{D}' \in \mathcal{D}$ is *too small* (e.g., it contains less than 25 domains at the first split) thus it is excluded from the final result;



(a) Mahalanobis distance ECDF for Alexa top 100,000 to identify λ and Λ.

(b) PCs of the Alexa top 100,000 domains and confidence thresholds.

Fig. 2. Non-DGA generated domains analysis

- a cluster of domains has its $\boldsymbol{M}$ matrix with *all the elements greater than zero*, meaning that the bipartite graph it represents is strongly connected;
- a cluster of domains *cannot be split further* by the DBSCAN algorithm with the value of ϵ set. In our experiments, we set ϵ to a conservative low value of 0.1, so to avoid the generation of clusters that contain domains that are not similar. Manually setting this value is possible because ϵ and the DBSCAN algorithm work on normalized features.

The final output of DBSCAN is $\mathcal{D}^\star = \{\mathbb{D}^1, \ldots, \mathbb{D}^R\}$. The domains within each $\mathbb{D}^r$ are similar among each other.

Dimensionality Reduction. The clustering algorithm employed has a space complexity of $O(|\mathbb{D}|^2)$. To keep the problem feasible we randomly split our dataset $\mathbb{D}$ into I smaller datasets $\mathbb{D}_i, i = 1, \ldots, I$ of approximately the same size, and cluster each of them independently, where I is the minimum value such that a space complexity in the order of $|\mathbb{D}_i|^2$ is affordable. Once each $\mathbb{D}_i$ is clustered, we recombine the I clustered sets, $\mathcal{D}_i^\star = \{\mathbb{D}^1, \ldots, \mathbb{D}^{R_i}\}$, onto the original dataset $\mathbb{D}$. Note that each $\mathbb{D}_i$ may yield a different number R_i of clusters. This procedure is very similar to the map-reduce programming paradigm, where a large computation is parallelized into many computations on smaller partitions of the original dataset, and the final output is constructed when the intermediate results become available. We perform the recombination in the following post-processing phase, which is run anyway, even if we do not need any dimensionality reduction (i.e., when $I = 1$, or $\mathbb{D}_1 \equiv \mathbb{D}$).

Clustering Post Processing. We post process the set of clusters of domains $\mathcal{D}_i^\star, \forall i$ with the following **Pruning** and **Merging** procedures. For simplicity, we set the shorthand notation $\mathbb{A} \in \mathcal{D}_i^\star$ and $\mathbb{B} \in \mathcal{D}_j^\star$ to indicate any two sets of domains (i.e., clusters) that result from the previous DBSCAN clustering, possibly with $i = j$.

Pruning. Clusters of domains that exhibit a nearly one-to-one relation with the respective IPs are considered unimportant because, by definition, they do not reflect the concept of DGA-based C&Cs (i.e., many domains, few IPs). Thus, we filter out the clusters that are flat and show a pattern-free connectivity in their bipartite domain-IP representation. This allows to remove "noise" from the dataset. Formally, a cluster $\mathbb{A}$ is removed if $\frac{|\mathrm{IPs}(\mathbb{A})|}{|\mathbb{A}|} > \gamma$, where γ is a threshold that is derived automatically as discussed in §5.

Merging. Given two independent clusters $\mathbb{A}$ and $\mathbb{B}$, they are merged together if the intersection between their respective sets of IPs is not empty. Formally, $\mathbb{A}$ and $\mathbb{B}$ are merged if $\mathrm{IPs}(\mathbb{A}) \cap \mathrm{IPs}(\mathbb{B}) \neq \emptyset$. This merging is repeated iteratively, until every combination of two clusters violates the above condition.

The outcome of the post-processing phase is thus a set of clusters of domains $\mathcal{E} = \{\mathbb{E}^1, \ldots, \mathbb{E}^Q\}$ where each $\mathbb{E}^q$ (1) exhibits a domain-to-IP pattern and (2) is disjointed to any other $\mathbb{E}^p$ with respect to its IPs. In conclusion, each cluster $\mathbb{E}$ contains the DGA-generated domains employed by the same botnet backed by the C&C servers at IP addresses $\mathrm{IPs}(\mathbb{E})$.

4.3 Step 3: Fingerprinting

The clusters identified with the previous processing are used to extract fingerprints of the DGAs that generated them. In other words, the goal of this step is to extract the invariant properties of a DGA. We use these fingerprints in the **Detection** module to assign labels to previously unseen domains, if they belong to one of the clusters. Given a generic cluster $\mathbb{E}$, corresponding to a given DGA, we extract the following cluster models:

- **CM1: C&C Servers Addresses** defined as $\mathrm{IPs}(\mathbb{E})$.
- **CM2: Length Range** captures the length of the shortest and longest domain names in $\mathbb{E}$.
- **CM3: Character Set** captures which characters are used during the random generation of the domain names, defined as $C := \bigcup_{e \in \mathbb{E}} \mathrm{charset}(p_e)$, where p_e is the chosen prefix of e.
- **CM4: Numerical Characters Ratio Range** $[r_m, r_M]$ captures the ratio of numerical characters allowed in a given domain. The boundaries are, respectively, the minimum and the maximum of $\frac{\mathrm{num}(p_e)}{|p_e|}$ within $\mathbb{E}$, where $\mathrm{num}(p_e)$ is the number of numerical characters in the chosen prefix of e.
- **CM5: Public Suffix Set** The set of eTDL employed by the domains in $\mathbb{E}$.

To some extent, these models define the aposteriori linguistic features of the domains found within each cluster $\mathbb{E}$. In other words, they define a model of $\mathbb{E}$.

4.4 Detection Module

This module receives a previously unseen domain d and decides whether it is a automatically generated by running the **Filtering** step with a loose threshold

λ. If d is automatically generated, it is matched against the fingerprints of the known DGAs on the quest for correspondences. In particular, we first select the candidate clusters $\{\mathbb{E}\}$ that have at least one IP address in common with the IP addresses that d pointed to: $\text{IPs}(d) \cap \text{IPs}(\mathbb{E}) \neq \emptyset, \forall \mathbb{E}$. Then, we select a subset of candidate clusters such that have the same models **CM1–5** of d. Specifically, the length of the chosen prefix of d, its character set, its numerical characters ratio, and the eTLD of d must lie within the ranges defined above. The clusters that survive this selection are chosen as the labels of d.

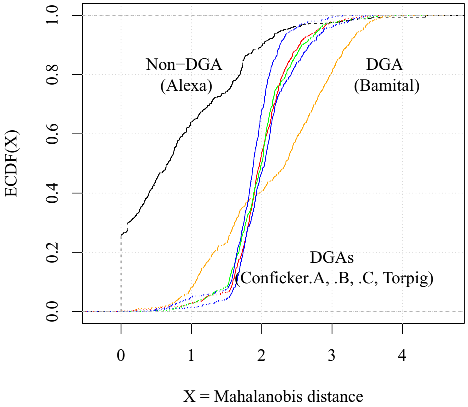
5 Experimental Evaluation

Validating the results of the Phoenix is challenging, because it produces novel knowledge. Therefore, we first validate the internal components of each module (e.g., to verify that they do not produce meaningless results and to assess the sensitivity of the parameters), and then we validate the whole approach using contextual information, to make sure that it produces useful knowledge with respect to publicly available information.

5.1 Evaluation Dataset and Setup

The **Discovery** module of Phoenix requires a feed of recursive DNS traffic and a reputation system that tells whether a domain is generally considered as malicious. For the former data source, we obtained access to the SIE framework (`dnsdb.info`), which provides DNS traffic data shared by hundreds of different network operators. We obtained traffic for about 3 months, totaling around 100B DNS requests and 4.8M distinct domain names. Differently from previous work, this type of traffic is privacy preserving and very easy to collect. For the latter data source we used the Exposure [6] blacklist, which included 107, 179 distinct domains as of October 1st, 2012.

Differently from previous work, we used DGA-generated domains merely as a ground truth for validation, not for bootstrapping our system before run time. More precisely, to validate the components of Phoenix we relied on ground truth generated by publicly available implementations of the DGAs used by Conficker [10] and Torpig [19], which have been among the earliest and most widespread botnets that relied on DGAs for C&C communication. Conficker's DGA is particularly challenging because it uses non-guessable seeds. With these DGAs we generated five datasets of domains, which resemble (and in some cases are equivalent to) the domains generated by the actual botnets: 7500, 7750 and 1,101,500 distinct domains for the **Conficker.A**, **Conficker.B** and **Conficker.C** malware, respectively, and 420 distinct domains for the **Torpig** dataset. Moreover, we collected the list of 36,346 domains that Microsoft claimed in early 2013 to be related to the activity of **Bamital** (`http://noticeofpleadings.com/`). We used a 4-core machine with 24GB of physical memory. Any experiment required execution times in the order of the minutes.



Family	$d_{Mah} > \Lambda$	$d_{Mah} > \lambda$
	Pre-cluster	Recall
Conficker.A	46.5%	93.4%
Conficker.B	47.2 %	93.7%
Conficker.C	52.9 %	94.8%
Torpig	34.2%	93.0%
Bamital	62.3%	81.4%

Fig. 3. Mahalanobis distance ECDF for different datasets (left), and pre-clustering selection and recall (right)

5.2 Discovery Validation

Step 1: Filtering. This filter is used in two contexts: by the **Discovery** module as a pre-clustering selection to recognize the domains that appear automatically generated within a feed of malicious domains, and by the **Detection** module as a pre-labeling selection. For pre-clustering, the strict threshold Λ is enforced to make sure that no DGA-looking domains pass the filter and possibly bias the clustering, whereas for pre-labeling the loose threshold λ is used to allow more domains to be labeled. The **Labeler** will eventually filter out the domains that resemble no known DGA. We test this component in both the contexts against the datasets of **Conficker**, **Torpig** and **Bamital** (never seen before).

The filter, which is the same in both the contexts, is best visualized by means of the ECDF of the Mahalanobis distance. Fig. 3 shows the ECDF from the datasets, compared to the ECDF from the Alexa top 100,000 domains. The plot shows that each datasets of DGA and non-DGA domains have different distribution: This confirms that our linguistic features are well suited to perform the discrimination. Indeed, the figure shows that each DGA dataset has a distinctive distribution, thus their DGAs are also different. On the other hand **Conficker** and **Torpig**'s DGAs have similar linguistic characteristics, although not identical. Then, we verify which fraction of domains passes the filter and reaches the **Clustering** (Λ) step or the **Labeler** (λ). The results obtained are reported in the first column of the table in Fig. 3 and show that roughly half of the domains would not contribute to the generation of the clusters: The conservative settings ensure that only the domains that exhibit the linguistic features more remarkably are used for clustering. Ultimately, most of the true DGA domains will be labeled as such before reaching the **Labeler**. Overall, Phoenix has a recall of 81.4 to 94.8%, which is remarkable for a non-supervised and completely automatic approach that requires no training.

Cluster 6 (Bamital)

```
50e7f66b0242e579f8ed4b8b91f33d1a.co.cc
bad61b6267f0e20d08154342ef09f152.co.cc
62446a1af3f85b93f4eef982d07cc492.co.cc
0d1a81ab5bdfac9c8c6f6dd4278d99fb.co.cc
f1dad9a359ba766e9f5ec392426ddd30.co.cc
295e2484bddd43bc43387950a4b5da16.co.cc
501815bd2785f103d22e1becb681aa48.co.cc
341af50eb475d1730bd6734c812a60a1.co.cc
49b24bf574b7389bd8d5ba83baa30891.co.cc
a7e3914a88e3725ddafbbf67444cd6f8.co.cc
```

Cluster 9 (Palevo via PushDo)

```
7cj1b.cn ff88567.cn  ef44ee.cn
fwjp0.cn  0bc3p.cn    9i230.cn
3dcyp.cn azeifko.cn fyyxqftc.cn
      hfju38djfhjdi3kd.cn
```

Cluster 10 (Palevo via PushDo)

```
ewn.net wyp.net ews.net kpk.net
khz.net uon.org lxx.net kxc.com
yhv.com nrl.net
```

Cluster 11 (Conficker)

```
  byuyy.biz   jbkxbxublgn.biz
  kpqzk.org     tcmsrdm.org
lvzqxymji.org  fbhwgmb.info
aeyyiujxs.org psaehtmx.info
 vdrmgyxq.biz    mmdbby.biz
```

Fig. 4. A representative example of a clustering obtained during our evaluation

In the pre-clustering phase, our system filtered out 34–62% of malicious, yet non-DGA domains. This ensures that the clusters are not "poisoned" with such domains, thus creating robust, conservative models.

Step 2: Clustering. We ran PHOENIX on our dataset and, after the first run of the DBSCAN clustering, we obtained a clustering for which we provide an excerpt in Fig. 4 (see [17] for full details). We can see that the clusters belonging to each botnet is profoundly different from a linguistic point of view. Interestingly, the clustering is not based on IP features, not linguistic features: This confirms that using linguistic features for first filtering non-DGA domains and then IP-based features to cluster them lead to clusters that reflect the actual botnet groups.

Reality Check. We searched for contextual information to confirm the usefulness of the clusters obtained by running PHOENIX on our dataset. To this end, we queried Google for the IP addresses of each cluster to perform manual labeling of such clusters with evidence about the malware activity found by other researchers.

We gathered evidence about a cluster with 33,771 domains allegedly used by Conficker (see also Fig.5 in [17]) and another cluster with 3870 domains used by Bamital. A smaller cluster of 392 domains was assigned to SpyEye (distributed through PushDo, `https://blog.damballa.com/archives/1998`), and two clusters of 404 and 58 domains, respectively, were assigned to Palevo (distributed through PushDo). We found no information to label the remaining 6 clusters as related to known malware.

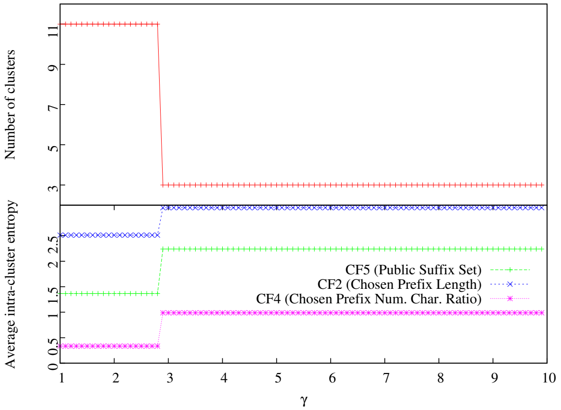


Fig. 5. Clustering sensitivity from parameter γ. By studying the number of clusters (top) and the average intra-cluster entropy over **CF2**, **4**, **5** (bottom), we can choose the best $\gamma \in (0, 2.8)$.

In conclusion, we successfully isolated domains related to botnet activities and IP addresses hosting C&C servers. From hereinafter we evaluate how well such isolation performs in general settings (i.e., not on a specific dataset).

Sensitivity from γ. We evaluated the sensitivity of the clustering result to the γ threshold used for cluster pruning. To this end, we studied the number of clusters generated with varying values of γ. A steady number of cluster indicates low sensitivity from this parameter, which is a desirable property. Moreover, abrupt changes of the number of clusters caused by certain values of γ can be used as a decision boundary to this parameter: Fig. 5 fixes that boundary at $\gamma = 2.8$.

We also assessed how γ influences the quality of the clustering to find safety bounds of this parameter within which the resulting clusters do not contain spurious elements. In other words, we want to study the influence of γ on the cluster models calculated within each cluster. To this end, we consider the cluster models for which a simple metric can be easily defined: **CM2 (Length Range)**, **CM4 (Numerical Characters Ratio Range)** and **CM5 (Public Suffix Set)**. A clustering quality is high if all the clusters contain domains that are uniform with respect to these models (e.g., each cluster contain elements with common public suffix set or length). We quantify such “uniformity” as the entropy of each model. As Fig. 5 shows, all the models reflect an abrupt change in the uniformity of the clusters around $\gamma = 2.8$, which corroborates the above finding.

In conclusion, values of γ outside $(0, 2.8)$ do not allow the clustering algorithm to optimally separate clusters of domains.

Correctness. Our claim is that the clustering can distinguish between domains generated by different DGAs by means of the representative IPs used by such DGAs (which are likely to be the C&C servers). To confirm this claim in a robust way, we evaluate the quality of the clustering with respect to features other than the IP addresses. In this way, we can show that our clustering tells different DGAs apart, regardless of the IP addresses in common. In other words, we show that our clustering is independent from the actual IP addresses used by the botnets but it is capable of recognizing DGAs in general.

To this end, we ignore **CM1** and calculate the models **CM2-5** of each cluster and show that they are distributed differently between any two clusters. We quantify this difference by means of the p-value of the Kolmogorov-Smirnov (KS) statistical test, which tells how much two samples (i.e., our **CM2-5** calculated for each couple of clusters) are drawn from two different stochastic processes (i.e., they belong to two different clusters). p-values toward 1 indicate that two clusters are not well separated, because they comprise domains that are likely drawn from the same distribution. On the other hand, p-values close to zero indicate sharp separation. The results confirm that most of the clusters are well separated, because their p-value is close to 0. In particular 9 of our 11 clusters are highly dissimilar, whereas two clusters are not distinguishable from each other (Clusters 2 and 4). From a manual analysis of these two clusters we can argue that a common DGA is behind both of them, even if there is no strong evidence (i.e., DNS features) of this being the case. Cluster 2 include domains such as `46096.com` and `04309.com`, whereas two samples from Cluster 4 are `88819.com` and `19527.com`. The actual p-values obtained in this experiments are detailed in [17].

5.3 Detection Evaluation

We want to evaluate qualitatively how well the **Detection** module is able to assign the correct labels to previously unseen suspicious domains. To this end, we first run the **Discovery** module using the historical domain-to-IP relations extracted from the SIE database for those domains indicated as generically malicious by the malicious domain filter (which is Exposure blacklist in our case). Once this module produced the clusters, we validated the outcome of the **Detection** against a never-seen-before (random) split of the same type of data.

This means that, given an unseen domain, which matches any cluster model, Phoenix generates novel knowledge by adding such a domain to the right cluster, thus effectively assigning a "threat name" to that domain. Domains that do not match any cluster model are not reported. The quality of the linguistic features and cluster models clearly affect the false negative rate, because they are conservative: More relaxed features and cluster models that still maintain a low degree of false negatives are focus of our ongoing research. The result of the **Detection** is a list of previously unseen domains, assigned to a cluster (i.e., a DGA). Some examples of previously unseen domains are depicted in Fig. 6 along with some samples of the clusters where they have been assigned to.

These examples show that Phoenix is capable of assigning the correct cluster to unknown suspicious domains. Indeed, despite the variability of the eTLD, which is commonly used as anecdotal evidence to discriminate two botnets, our system correctly models the linguistic features and the domain-to-IP historical

Previously unseen domains			
hy613.cn	5ybdiv.cn	73it.cn	39yq.cn
69wan.cn	hy093.cn	08hhwl.cn	hy267.cn
hy673.cn	onkx.cn	xmsyt.cn	fyf123.cn
watdj.cn	dhjy6.cn	algxy.cn	g3pp.cn

Previously unseen domains			
dky.com	ejm.com	eko.com	blv.com
efu.com	elq.com	bqs.com	dqu.com
bec.com	dpl.com	eqy.com	dyh.com
dur.com	bnq.com	ccz.com	ekv.com

Cluster 9 (Palevo)

pjrn3.cn	3dcyp.cn	x0v7r.cn	0iwzc.cn
0bc3p.cn	hdnx0.cn	9q0kv.cn	4qy39.cn
5vm53.cn	7ydzr.cn	fyj25.cn	m5qwz.cn
qwr7.cn	xq4ac.cn	ygb55.cn	v5pgb.cn

Cluster 10 (Palevo)

uon.org	jhg.org	eks.org	kxc.com
mzo.net	zuh.com	bwn.org	khz.net
zuw.org	ldt.org	lxx.net	epu.org
ntz.com	cbv.org	iqd.com	nrl.net

Fig. 6. Labeling of previously unseen domains

relations and performs a better labeling. In the second case the domains were registered under .cn and share the same generation mechanism.

5.4 Intelligence and Insights

In this section, we describe two use cases of the **Intelligence and Insights** module, which provides the analyst with valuable knowledge from the outputs of the other modules. The correctness of the conclusions drawn from this module is predicated on the correctness of the two upstream modules, already discussed in prevoius sections.

Unknown DGA Recognition from Scarce Data. Our system is designed to automatically label the malicious domains related to botnet activities. This is done by using the information of the DNS traffic related to them. Interestingly, some conclusions can be drawn on previously unseen domains even in the unlucky case that such information is missing (i.e., when no DNS data is available).

On Feb 9th, 2013 we received, via a vetted security mailing list, an inquiry by a group of researchers. They had found a previously unseen list of DGA-generated domains that resembled no known botnet. Such list was the only information that they provided us with. Phoenix correctly labeled these domains with the fingerprints of a Conficker cluster. This allowed the researchers to narrow down their investigation.

In conclusion, starting from the sole knowledge of a list of malicious domains that Phoenix had never seen before, we discovered that, according to our datasets, the only DGA in our dataset able to produce domains with that linguistic features was the DGA associated with **Conficker**.

Time Evolution. Associating DGA domains to the activity of a specific botnet allows to gather further information (e.g., track the botnet evolution) by using the DGA fingerprints as a "lookup index" to make precise queries.

For instance, given a DGA fingerprint or a sample domain, we can select the domains of the corresponding cluster $\mathbb{E}_{DGA}$ and partition this set at different granularity (e.g., IPs or ASs) by considering the *exact* set of IPs (or ASs) that

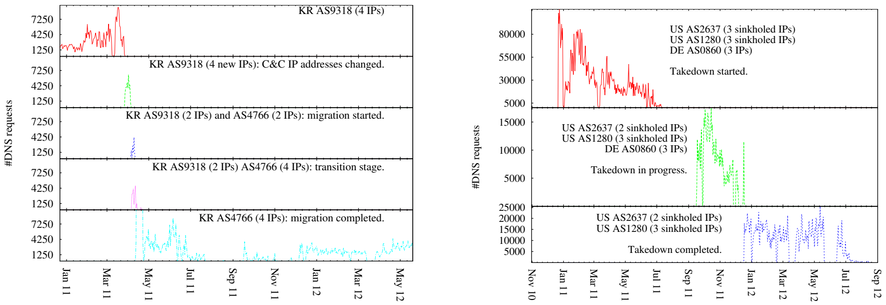


Fig. 7. Bamital (left): Migration of C&C from AS9318 to AS4766. **Conficker (right)**: Evolution that resembles a C&C takedown: the C&C had 3 IPs in AS0860 and 3 sinkholed IPs in AS2637

they point to. Given the activity that we want to monitor, for instance, the DNS traffic of that botnet, we can then plot one time series for each partition. In our example, we count the number of DNS requests seen for the domains in that partition at a certain sampling frequency (e.g., daily). The analysis of the stacked time series generated allows to draw conclusion about the behavior over time of the botnet. Fig. 7 shows the case of (a) a migration (the botmaster moved the C&C servers from one AS to another) followed by (b) a load balancing change in the final step (the botmaster shut down 2 C&C servers thus reducing the load balancing).

In a similar vein, Fig. 7 shows an evolution that we may argue being a takedown operated by security defenders. In particular, at the beginning the botnet C&C backend was distributed across three ASs in two countries (United States and Germany). Armed with the knowledge that the IPs in AS2637 and AS1280 are operated by computer security laboratories, we discover that this “waterfall” pattern concludes into a sinkhole. Without knowledge of the sinkholed IPs, we can still argue that the C&C was moved to other ASs.

The aforementioned conclusions were drawn by a semi-automatic analysis and can be interpreted and used as novel intelligence knowledge. The labels of the DGAs produced by Phoenix were fundamental to perform this type of analysis.

6 Limitations

Despite the good results, Phoenix has some limitations. Previous work leveraged NXDOMAIN responses to identify those DGA-generated domains that the botmaster did not register yet. This allows early detection of DGA activities, because the bots yield overwhelming amounts of NXDOMAIN replies. Our system, instead, requires *registered* domains to function. Therefore, it is fed with data that takes slightly longer collection periods. This results in a less-responsive detection of previously unseen DGAs. The advantage is that, differently from previous work, we can fingerprint the DGAs and, more importantly, we lift the observation point such that Phoenix is easier to adopt. Indeed, we believe that not using NXDOMAIN replies represents a strength of our work, as it makes our system profoundly different from previous work in ease of deployment and testing under less-constraining requirements.

The linguistic features computed on the domain names, to decide whether they are automatically generated or not, capture the likelihood that a given domain targets English-speaking users. Taking into account different languages, possibly featuring totally different sounds like Chinese or Swedish, as well as different encondings, such as UTF8, would pose some challenges. In particular, computing language-independent features with a multilingual dictionary would flatten the underlying distributions, rendering the language features less discriminant. To tackle this limitation, a possible solution consists in inferring the linguistic target of a given domain (e.g., via TLD analysis or whois queries) so to evaluate its randomness according to the correct dictionary.

Future DGAs may attempt to evade our linguistic features by creating *pronounceable* domains. Besides the fact that, to the best of our knowledge, no such DGAs exist, creating *large amounts* of pronounceable domains is difficult: Such DGAs would have a narrow randomization space, which violates the design goals of domain flux [10, 19].

7 Related Work

The *idea* of using linguistic features per se is not novel. However, existing approaches are based on supervised learning and make assumptions on how domains should be grouped before processing. Yadav et al. [21, 22] leverage the randomization of DGA-generated names to distinguish them from non-DGA ones by means of linguistic features bi-grams computed over domain *sets*, which are then classified as sets of DGA- or non-DGA-related. The work explores different strategies to group domain in sets before feeding them to the classifier. Our work is different from these approaches because we require no labeled datasets of DGA domains to be bootstrapped, thus it is able to find sets of DGA domains with no prior knowledge. Moreover, our system classifies domains one by one, without the necessity of performing error-prone apriori grouping.

Phoenix differentiates from the approaches that model DGAs as a mean to detect botnet activity by the type of knowledge that it produces and by the less-demanding requirements. Perdisci et al. [15] focused on domains that are malicious, in general, from the viewpoint of the victims of attacks perpetrated through botnets (e.g., phishing, spam, drive-by download). Moreover, the detection method of [15] is based on supervised learning. Neugschwandtner et al. [12] proposed a system that detects malware failover strategies with techniques based on multi-path exploration. Backup C&C servers and DGA domains are unveiled through simulated network failures, leading to new blacklists. Although promising, the approach requires the availability of malware samples. Differently from [12], we only recursive-level passive DNS traffic.

Phoenix differentiates from the approaches that leverage features of DNS packets to find new malicious domains by the type of new knowledge inferred and by the less-demanding learning technique. For example, [6] is a passive DNS analysis technique to detect domains associated with malicious activities, including botnet C&C. The main difference is that Phoenix focuses exclusively on DGAs rather than inferring a domain's maliciousness. Instead of training a classifier on malicious domains, we calculate thresholds for our filters based on *benign*—or, at least, human-generated—domains. Systems like [6] and [1] rely on local recursive DNS. Instead, [2] analyzes DNS traffic collected at the upper DNS hierarchy with new features such as the requester diversity, requester profile and resolved-IPs reputation. As the authors notice, the approach is ineffective on DGA-generated domains, because of their short lifespan, whereas we have showed extensively that Phoenix can detect and, more importantly, label, previously unknown DGA domains. Bilge et al. [5] proposed DISCLOSURE, a system that detects C&C communications from NetFlow data analysis. Using NetFlow data overcomes the problems of large-scale traffic collection and processing. However, Disclosure discovers domains involved in C&C communications, not necessarily DGAs.

Other approaches leverage that DGA-based malware yield disproportionately large numbers of NX responses. Yadav and Reddy [20] extend [22] and introduce NXDOMAINs to speedup the detection of DGA-generated domains: *registered* DGA-generated domains are recognized because they are queried by any given client after a series of NXDOMAIN responses. The work differs from ours substantially, mainly because it requires DNS datasets that include the IP addresses of the querying clients. Moreover, the approach seems fragile on sampled datasets, which is a required step when dealing with high-traffic networks. To some extent, our work is complementary to the use of NXDOMAINs, which can be used to provide early, yet not very explanatory, warnings. Our system compensates for this lack through the intelligence and insights module.

8 Conclusion

In addition to telling DGA- and non-DGA-generated domains apart using a combination of linguistic and IP-based features, Phoenix characterizes the DGAs behind them, and finds groups of DGA-generated domains that are representative of the respective botnets. As a result, Phoenix can associate previously unknown DGA-generated domains to these groups, and produce novel knowledge about the evolving behavior of each tracked botnet. We improve the linguistic features proposed in previous work and combine them with other features. We also calculate fingerprints of the domains identified by Phoenix as belonging to a group of "similar" domains. Contrarily to the existing methods based on NX domains, our approach does not rely on clients' IPs, is not affected by NAT or DHCP, and requires no specific deployment contexts.

We successfully used Phoenix in real-world settings to identify a list of suspicious domains as belonging to a live botnet (based on Conficker.B). We believe that, in addition to the comprehensive evaluation, this latter fact proves Phoenix's practicality and effectiveness.

References

[1] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for dns. In: USENIX Security (2010)

[2] Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N., Dagon, D.: Detecting malware domains at the upper DNS hierarchy. In: USENIX Security, vol. 11 (2011)

[3] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX Security, USENIX Association (August 2012)

[4] Bailey, T.M., Hahn, U.: Determinants of wordlikeness: Phonotactics or lexical neighborhoods? Journal of Memory and Language 44(4), 568–591 (2001)

[5] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: ACSAC. ACM (2012)

[6] Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive DNS analysis. In: NDSS (2011)

[7] Han, J., Kamber, M.: Data mining: concepts and techniques. Morgan Kaufmann (2006)

[8] Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and detecting fast-flux service networks. In: NDSS (2008)

[9] Jones, E., Oliphant, T., Peterson, P.: et al.: SciPy: Open source scientific tools for Python (2001), http://www.scipy.org/ (accessed: January 28, 2013)

[10] Leder, F., Werner, T.: Know your enemy: Containing conficker. The Honeynet Project, University of Bonn, Germany, Tech. Rep. (2009)

[11] Marinos, L., Sfakianakis, A.: ENISA Threat Landscape. Tech. rep., ENISA (2012)

[12] Neugschwandtner, M., Comparetti, P.M., Platzer, C.: Detecting malware's failover C&C strategies with Squeeze. In: ACSAC. ACM (2011)

[13] Newman, M.: Networks: an introduction. Oxford University Press (2010)

[14] Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: fluXOR: Detecting and monitoring fast-flux service networks. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 186–206. Springer, Heidelberg (2008)

[15] Perdisci, R., Corona, I., Giacinto, G.: Early detection of malicious flux networks via large-scale passive DNS analysis. IEEE Transactions on Dependable and Secure Computing 9(5), 714–726 (2012)

[16] Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., van Steen, M.: Prudent practices for designing malware experiments: Status quo and outlook. In: Security and Privacy (SP). IEEE (2012)

[17] Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Tracking and Characterizing Botnets Using Automatically Generated Domains. Tech. rep. (2013), http://arxiv.org/abs/1311.5612

[18] Scholes, R.J.: Phonotactic grammaticality. No. 50, Mouton (1966)

[19] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your botnet is my botnet: analysis of a botnet takeover. In: CCS. ACM (2009)

[20] Yadav, S., Reddy, A.L.N.: Winning with DNS failures: Strategies for faster botnet detection. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 446–459. Springer, Heidelberg (2012)

[21] Yadav, S., Reddy, A.K.K., Reddy, A., Ranjan, S.: Detecting algorithmically generated domain-flux attacks with dns traffic analysis. IEEE/ACM TON 20(5) (2012)

[22] Yadav, S., Reddy, A.K.K., Reddy, A.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: IMC. ACM (2010)