Identifying Suspicious Activities through DNS Failure Graph Analysis

Nan Jiang*, Jin Cao†, Yu Jin*, Li Erran Li†, Zhi-Li Zhang*
*Computer Science Dept., University of Minnesota †Bell Laboratories, Alcatel-Lucent
*{njiang,yjin,zhzhang}@cs.umn.edu †{cao,erranlli}@research.bell-labs.com

***Abstract*—As a key approach to securing large networks, existing anomaly detection techniques focus primarily on network traffic data. However, the sheer volume of such data often renders detailed analysis very expensive and reduces the effectiveness of these tools. In this paper, we propose a light-weight anomaly detection approach based on *unproductive* DNS traffic, namely, the failed DNS queries, with a novel tool – *DNS failure graphs*. A DNS failure graph captures the interactions between hosts and failed domain names. We apply a graph decomposition algorithm based on the tri-nonnegative matrix factorization technique to iteratively extract coherent co-clusters (dense subgraphs) from DNS failure graphs. By analyzing the co-clusters in the daily DNS failure graphs from a 3-month DNS trace captured at a large campus network, we find these co-clusters represent a variety of anomalous activities, e.g., spamming, trojans, bots, etc.. In addition, these activities often exhibit distinguishable subgraph structures. By exploring the temporal properties of the co-clusters, we show our method can identify new anomalies that likely correspond to unreported domain-flux bots.**

I. Introduction

The Internet Domain Name System (DNS) is a critical infrastructure service used by nearly every Internet application for locating various resources (e.g., web servers, mail servers, individual endhosts) specified by their (host) domain names. Typically, one endpoint first issues a DNS query to the DNS system to locate the other endpoint before any subsequent data transfer between the two communicating endpoints can commence, be it web downloading, email transfer, instant messaging, or a VoIP call placed on the Internet. A DNS query failure often signifies that the requested resource does not exist in the system when the query is issued. While such a failure may be caused by a mis-typed host name or URL by a human user or occasionally due to DNS misconfigurations by human operators [1], a large portion of DNS query failures can be attributed to other causes — as pointed out in several recent studies [2], [3], [4]. For instance, several anti-spam and anti-virus services employ DNS "overloading" to notify a querying host whether the requested domain name belongs to the blacklists they maintain (e.g., of email spam servers or reported attack sites). In particular, as shown in [3], many DNS query failures (termed "unproductive" DNS traffic) are caused by "suspicious" and malicious cyber activities, e.g., fast-flux web services, trojan malware and botnets [5], [6], [7], [8].

Inspired by these studies, in this paper we advance the notion of *DNS failure graphs* as an effective means for analyzing "unproductive" DNS traffic in a *systematic* manner and from a *network-wide* perspective, and for detecting and identifying (large-scale) suspicious and malicious cyber activities. A *DNS failure graph* is a bipartite graph consisting of domain names of failed DNS queries and hosts issuing such queries, with an edge between a domain name and a host issuing a (failed) DNS query for the name. Such a graph can be constructed using "unproductive" DNS traffic collected at one or multiple networks (or from any host on the Internet, if such data can be collected). The basic intuition behind this notion is that hosts infected by the same malware (e.g., belonging to the same botnet) usually query for the same, similar or otherwise correlated set of domain names, for instance, to locate the Command & Control (C&C) servers, malware hosting sites, stolen data storage servers, etc. To evade detection, the domain names used by these malicious activities often change frequently (i.e., in domain-flux[9], [10], [11]); those that do not flux frequently often are blacklisted and blocked after detection. Hence queries for these domain names frequently result in *correlated* failures, which manifest themselves as a *dense* subgraph in a DNS failure graph. Such dense subgraphs therefore capture the strong *interaction patterns* between a set of hosts and a set of domain names. This observation gives rise to a key research question that we address in this paper: *Can we effectively identify, differentiate and separate "subgraphs" that are likely corresponding to different types of anomalies (e.g., malware activities) based on the interaction patterns between hosts and domain names in a DNS failure graph?*

To answer this question, we utilize the DNS query data collected at several major DNS servers of a large campus network over a three-month period. Through systematic analysis of the "unproductive" DNS traffic contained in this three-month DNS query data, we find that while the DNS failure graphs (e.g., constructed using failed DNS queries each day) typically consist of a large number of isolated (connected) components, there often exist one or several "giant" connected components involving a large number of hosts and domain names. While these giant components are connected, they themselves appear to be composed of a number of more densely connected subgraphs. In other words, one cannot simply take each isolated component – especially when such a component is large and involves a significant number of hosts and domain names – as representing and corresponding to a single type of anomaly. We therefore apply a (statistical) graph decomposition technique, which extends the *tri-nonnegative matrix factorization (tNMF)* [12] algorithm, to recursively decompose a DNS failure graph and extract dense (bipartite)

978-1-4244-8643-4/10/$26.00 ©2010 IEEE

subgraphs, or *co-clusters*, representing strong and coherent interaction patterns. By analyzing their structural properties, we classify the resulting co-clusters into three categories: 1) a *host-star*, where a few hosts dominate by sending a large number of DNS queries; 2) a *DNS-star*, where a few domain names attract queries from many hosts; 3) a *bi-mesh*, where strong interaction patterns are observed between a group of hosts and a group of domain names. Using external data sources such as domain name blacklists, we find that most of the DNS-stars are caused by instances of trojan malware accessing blocked domain names. In comparison, the host-stars are primarily the artifacts of spamming activities involving queries for expired domain names of certain email servers. Most interestingly, many bi-mesh structures are found to be associated with bot activities, where the hosts infected by the same bots query a list of domain names that are likely those of C&C servers, malware hosting sites, and other suspicious resources.

We further characterize and distinguish the suspicious activities associated with these co-clusters by exploring their temporal properties and tracking their evolution over time. We find that a majority of the co-clusters are associated with a stable set of domain names, suggesting that the infected hosts in each co-cluster likely belong to a botnet with a list of hard-coded domain names for querying C&C and other servers. In contrast, we also find that several co-clusters are associated with a set of domain names that flux over time. Analyzing the patterns of domain names involved, the rate they are generated, and corroborating them with existing studies, we identify four of them belonging to several known domain-flux bots. The remaining ones have similar random-looking, but yet distinct domain name patterns; further, their domain name flux rates differ considerably from those of the known domain-flux bots. These observations lead us to believe that they are plausibly associated with domain-flux bots that are yet to be reported, and hence require further scrutiny.

Summary and Contributions. The main contributions of the paper are three-fold: i) we advance the notion of DNS failure graphs for network-wide analysis of "unproductive" DNS traffic; ii) we propose an extension of the tNMF graph decomposition method and demonstrate how it can be applied to extract dense subgraphs or co-clusters, which represent strong and coherent interaction patterns between hosts and domain names; and iii) we develop novel methods to systematically analyze, classify and track the structural and other properties of the extracted co-clusters and their evolution over time, and by corroborating with other data sources, deduce that the extracted co-clusters capture correlated DNS failures that are generally associated with same or similar types of anomalies such as malware or botnet activities.

Unlike many existing anomaly detection techniques which focus primarily on network traffic data – the sheer volume of such data often renders detailed analysis very expensive and reduces the effectiveness of these tools (e.g., too many false positives or negatives), our work provides an effective means to identify and detect large-scale exploits by analyzing and decomposing *unproductive* DNS traffic – much of which are "footprints" left by these exploits – from a network-wide perspective. Clearly, analyzing DNS failure queries alone is insufficient in detecting large-scale exploits; nonetheless, our DNS failure graph analysis can help winnow down and zero in on likely suspicious activities. Advanced anomaly detection and malware analysis techniques using network traffic data can then be effectively applied to these suspected malicious activities. In summary, our work adds a useful and complementary tool to the existing arsenal of techniques for detecting and combating large-scale exploits. We believe that it can be used as a "first-line" defense in identifying emerging threats that are constantly changing and evolving.

The remainder of the paper is organized as follows. We first discuss the related work in Section II. In Section III, we analyze the failed DNS queries and introduce the notion of DNS failure graphs. We then propose a co-clustering algorithm for decomposing DNS failure graphs into strongly connected subgraphs in Section IV. Section V presents the classification and interpretation of these dense subgraphs and their temporal properties are studied in Section VI. Finally, Section VII concludes the paper.

II. Related work

As mentioned earlier, our work is motivated by prior work such as [2] which first points out using DNS queries for detecting bots, [3] which employs a supervised machine learning method to classify different attacks using a combination of DNS query failures and network traffic data collected for individual hosts, and [4] which provides a systematic analysis and classification of DNS traffic. Building upon these earlier studies, our work puts forth a novel and effective methodology for *network-wide analysis* of unproductive DNS traffic via DNS failure graph decomposition, and demonstrates how the method can be used to identify and differentiate suspicious activities using *correlation between hosts and the failed DNS queries*. For instance, our analysis uncovers groups of hosts with correlated DNS query failures that differ from known domain-flux bots and are plausibly part of domain-flux or similar botnets that are yet to be reported. Compared with [2], [4], our method explores explicitly the correlation of failed DNS queries (with small traffic volume) for detecting network anomalies. Unlike [3], our method employs an unsupervised machine learning approach and thus does not require training data with expensive manual labels. Similar to our work, [13] uses co-occurrence relation among DNS queries to extend black domain name lists.

In addition to the study of unproductive DNS traffic, there is a rich literature regarding anomaly detection by monitoring "unwanted" traffic. Pang et. al. [14] study the traffic towards unallocated IP blocks (dark space). Similar approach has been applied for "trapping" unproductive traffic using honeynet [15]. Jin et. al. [16] characterize and classify the traffic towards temporally unassigned IP addresses(gray space). Similar to these existing works, we demonstrate in this paper that unproductive DNS traffic can also be used

TABLE I: Categories of failed DNS queries.

Type	Pct (%)	Examples	Description
DNS Overloading	32.37	anti-spam/anti-virus applications	spamcop.net, surbl.org
Server errors	28.01	unresolvable domain names in a server farm	crpkcmsaw00.bankofamerica.com
Misconfigurations	7.87	recursive DNS resolver	www.example.com.example.com
Typos	2.26	misspelling of domain names	googloe.com, encyclopiedea.net
Known Threats	2.08	blocked trojan/worm	g43gwef.com, antispyware2008xp.com
P2P	0.75	failures in P2P related applications	66bt.cn, zingking.com
Unknown	27.33	unknown failures	vuuewgkt.com, dehpydjsi.cn

to effectively detecting network anomalies, especially botnet behaviors.

Our work is also related to botnet detection. There have been quite a few studies focusing on p2p botnets [5] and spam botnets [6], [7], [8]. These works either manually inject bots into the botnets or use spamming behaviors to group bots from the same botnet. For instance, Holz et al [5] examines the details of the Storm Worm botnets. They proposes two approaches to mitigate the botnets. The first is called *eclipse* attack, a special form of the *sybil* attack. The goal is to separate a part of the P2P network from the rest. The second approach is called polluting attack, whose goal is to "overwrite" the content previously published under a certain key. These attacks are specific to P2P botnets, and they do not apply as general botnet detection techniques. Many spamming botnets detection methods [6], [7], [8] make use of similar spamming behaviors to detect and classify bots from different botnets. However, these measurements are more expensive and hard to obtain compared to the DNS traffic. Moreover, our method by passively monitoring failed DNS queries is a more light-weight approach than the existing methods, given the much lower volume of the DNS traffic.

There have been studies focusing on individual botnets which maintain connections between the bots and the C&C servers using techniques like domain-flux [9], [11], [10] or fast-flux [17], [18]. These works rely on capturing bot instances and identify bot signature or the domain name generation (DGA) algorithms via reverse engineering. These methods are in general very expensive and require strong prior knowledge on the specific botnets. Hence, unlike our work, these methods do not generalize.

III. DNS Traffic and Failure Graphs

In this section, we advance the notion of *DNS failure graphs*, which capture the patterns that hosts query for non-existing domain names. We first briefly describe the datasets used in this paper. We then provide an overview analysis of failed DNS queries in term of their plausible causes and formally define DNS failure graphs. At the end of the section, we present an analysis of the properties of DNS failure graphs and demonstrate the community structures (or densely connected subgraphs) in DNS failure graphs.

Datasets. Our study utilizes the DNS data collected at a large university campus network over a 3-month period (from Jan. 2009 to Mar. 2009). The network contains around $20K$ hosts, with IP addresses assigned either statically (e.g., lab machines, web or mail servers) or dynamically (e.g., hosts on residential dormitory networks or wireless LANs). The collected DNS dataset contains DNS requests and responses from all hosts within the campus network for locating resources outside the campus network. The data is in the format of packet traces collected using TCP dump. For DNS requests, we have the information of (anonymized) hosts who initiate the queries and the target domain names. For DNS responses, we have access to resolved IP addresses and associated response codes (if any). We focus on *type A* DNS requests only, which queries for the IPv4 address(es) associated with a domain name. We refer to the DNS queries for which the DNS responses contain a response code other than "NOERROR" as *failed DNS queries*. Each day approximately 2 million DNS queries are captured, in which around $300K$ are failed DNS queries.

A. Analysis of Failed DNS Queries

We first investigate the plausible causes for such a large number of failed DNS queries in the network by examining patterns in the failed DNS queries as well as utilizing other data sources. Table I shows a sample classification of the failed DNS queries on 01/05/2009. We observe that a large portion of failed DNS queries are due to the so-called "overloaded traffic" [4], where several anti-spam and anti-virus services employ DNS to notify a querying host whether the requested domain name belongs to the blacklists they maintain (e.g., of email spam servers or reported attack sites). We observe that this type of failed DNS queries involves only a small number (fewer than 20) of hosts, mostly email servers for spam filtering purpose. Server error is the second major contributor to the failed DNS queries. Such failed DNS queries are caused by one or a few domain names related to a popular web service that are temporarily unresolvable. DNS misconfigurations such as a query for *www.example.com.example.com* (such "recursive domain names" are likely due to Windows default DNS suffix configured at client machines) account for 7.87% of all the failed DNS queries, while DNS typos, which are likely caused by users mistyping a few alphabetics of the desired domain names, account for 2.26%.

For the remaining failed DNS queries, we look up the target domain names in each failed query in a number of auxiliary data sources, including various blacklists [19], security logs [20], botnet related domain names obtained via reverse engineering [21], and information obtained by googling the Internet [22]. If a target domain name is used by a worm/trojan and blacklisted, we attribute the failed DNS query as *Known Threats*. We find that 2.08% of the failed DNS queries belong to this category. Another 0.75% of the failed DNS queries can be attributed to hosts participating in p2p activities, as the target domain names are associated with p2p applications and services found on-line. Finally, we cannot properly attribute the causes for the remaining 27.33% of the failed DNS queries using various on-line sources mentioned above, and thus classify them as *Unknown*. We manually inspect these *Unknown* domain names and find that most of these targeted domain names contain random-looking strings with distinct patterns. As we shall see later in Section VI, most of them are likely associated with suspicious activities, e.g., unreported domain-flux botnet activities.

B. DNS Failure Graphs and Properties

So far, we identify potential threats in "unproductive" DNS traffic by matching the target domain names in failed DNS queries against data sources of known security threats. However, such a method is rather time-consuming, whose effectiveness hinges highly on the availability of useful external data sources. By its very nature, this method cannot be used to detect *emerging* threats that are yet to be discovered and reported. As shown in Table I, a significant portion (27%) of failed DNS queries cannot be attributed to *known* threats. The large majority of these failed DNS queries contain domain names that are suspicious looking and are unlikely to represent "legitimate" resources on the Internet, we have little information regarding them. Hence we are interested in an *automatic* method for identifying suspicious activities behind these failed DNS queries. This motivates us to develop the *DNS failure graph analysis* technique presented in this paper. Our basic idea is that suspicious activities are often reflected as strong correlations between hosts and failed domain names. This is because hosts infected by the same malware or participating in the same activity tend to access similar non-existing domain names and hence generate same failed DNS queries. Using a (bi-partite) DNS failure graph to capture the interactions between hosts and domain names, a strong correlation between hosts and the DNS query failures is reflected directly by a densely connected subgraph in the corresponding DNS failure graph. Thus the problem of identifying suspicious activities can be casted as the problem of extracting strongly connected subgraph components from the DNS failure graph.

Before we perform the DNS failure graph analysis, we first "cleanse" the failed DNS queries by filtering the ones that are attributable to "normal" network activities such as DNS overloading, server errors and misconfigurations. We note that we have developed a heuristic cleansing procedure to automatically filter these "normal" DNS query failures. For example, we filter overloaded DNS query failures by matching the responders of these queries against a list of known anti-spam/anti-malware sites, and adopt a similar approach as proposed in [2] for filtering failed DNS queries due to server errors. Due to space limitation, we do not provide the detailed heuristics used here. Note that we do not automatically filter failed DNS queries involving p2p activities, partly because they are hard to filter automatically. More importantly, many p2p applications or services are sometimes abused by malware activities; some of them appear suspicious on their own. Since our objective is to use failed DNS queries to identify potentially suspicious activities, we perform this cleansing step mainly to reduce the amount of data used in the DNS failure graph analysis. The cleansing procedure is fairly conservative in the sense that we only filter failed DNS queries that can be confidently attributed to *normal* network activities. In fact, as will be evident in our DNS failure graph analysis later, most failed DNS queries due to normal activities are well separated from suspicious ones. Hence this cleansing procedure in general does not affect the effectiveness of our DNS failure graph analysis technique.

We now formally define *DNS failure graphs*: Given an observation period T (in our experiments, we always choose T = 1 day to maximize the amount of correlations observed and eliminate the effect of IP address churns [23]), let $\mathcal{H}$ denote the set of hosts (IP addresses) making at least one failed DNS query, and $\mathcal{D}$ be the set of (unique) domain names in the failed queries. A *DNS failure graph* is a bipartite graph $\mathcal{G} := \{\mathcal{H} \times \mathcal{D}, \mathcal{E}\}$, where an edge $e = (h, d)$ exists between a host $h \in \mathcal{H}$ and a domain name $d \in \mathcal{D}$, i.e., $(h, d) \in \mathcal{E}$, if and only if host h makes at least one failed DNS query[1] for d during the observation time period T. Given this definition, we construct daily DNS failure graphs (i.e., T = 1 day) using our datasets. We observe that in general there are roughly 2,000 hosts connecting to around 3,000 failed domain names each day. Each daily DNS failure graph is often composed of 1000 or more *isolated* components (subgraphs): each component is connected, but there is no edge connecting any two (connected) components (i.e., the components are isolated from each other). Despite the large number of isolated components – a large majority of them are small, there exist a few components that are significantly larger than the others. We measure the size of each component in terms of the percentage of hosts covered by the component out of all hosts. Fig. 1 shows the sizes of the largest components over a two-week period (from 01/05/2009 to 01/18/2009), where the solid curve in the figure represents the size of the largest components in the daily DNS failure graphs; for comparison, the dotted curve represents the size of the largest component in the *cumulative* DNS failure graphs constructed by varying T from 1 day up to the entire two weeks. We see that the size of the largest component in the daily DNS failure graphs ranges from 14% to 37%. As the observation period T expands from 1 day up to the entire two weeks, more hosts (77% in the entire two weeks) are included in the largest component; the big jump in the curve is caused by two large components (in two different days) connected by a single host.

Despite their large sizes, these connected components are comprised of many loosely connected (e.g., via a few edges) subgraphs, each of which is more densely connected. We use the largest component in the daily DNS failure graph on 01/05/2009 to illustrate this point by visualizing it using

[1] We remark that in this paper we consider the DNS failure graphs to be unweighted, representing the absence/presence of a certain DNS query. However, our method can be readily extended to weighted DNS traffic graphs, where the weight of an edge (h, d) can be used to represent, e.g., the number of failed queries from host h for d.

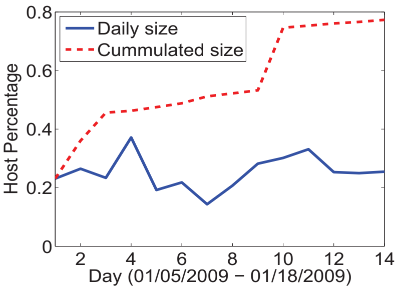


Fig. 1: Size of the largest DNS failure subgraph over time.

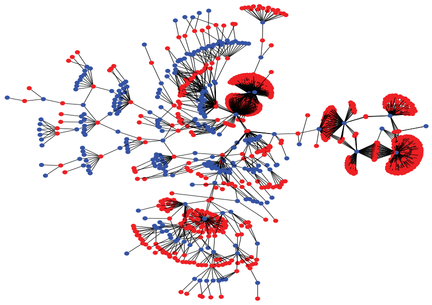


Fig. 2: The largest DNS failure subgraph from 01/05/2009.

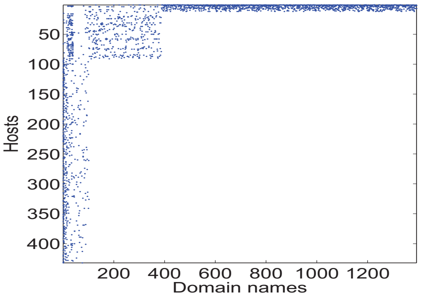


Fig. 3: Block structures after row and column rotations.

Graphviz [24], as shown in Fig. 2, where the blue nodes and red nodes represent hosts and domain names, respectively (For clarity of visualization, we have randomly removed 60% of nodes with degree 1 in Fig. 2). Clearly, this largest connected component contains several dense subgraphs that are loosely connected via a few edges. These dense graphs imply that there exist strong *correlated behaviors* ("community structures" in social network analysis jargons) among the hosts in these dense subgraphs: the strong correlations manifest in the failed domain names they query; in other words, there are *strong interaction patterns* that connect the set of hosts and the set of domain names they collectively query.

To further illustrate these "community structures," we represent the same graph in Fig. 2 using its adjacency matrix $A = [a_{ij}]$. The rows and columns of A represent the hosts ($\mathcal{H}$) and the domain names ($\mathcal{D}$), respectively; entry $a_{ij} = 1$ if edge $(h_i, d_j) \in E$, and $a_{ij} = 0$ otherwise. We rotate the rows and columns in A to best reflect the "community structures" in the graph. We plot the rotated A in Fig. 3, where dots represent those non-zero entries in A. The "community structures" (dense subgraphs) in the graph are now visible as "blocks" in A. Further, we see that there are several types of "community" or "block" structures: some contain a small number of hosts but a large number of domain names, other contain a large number of hosts but a smaller number of domain names, and yet other contain both relatively large numbers of hosts and domain names. These different interaction patterns between the hosts and domain names suggest that the hosts involved are likely engaging in different kinds of suspicious activities. These visual analyses suggest that the largest connected component can be further *decomposed* into dense subgraphs, which more likely correspond to correlated behaviors. These dense subgraphs are connected by a few weak links or random edges which are shown as the light area in Fig. 3. Due to the existence of these weak links, we cannot simple treat each subgraph as a single activity. Instead, we need to extract these dense graph components or "communities" to separate different activities.

IV. Decomposing DNS failure graphs

In this section, we present an algorithm for automatically decomposing, and extracting dense subgraphs from, DNS failure graphs. This algorithm extends tNMF-based graph decomposition technique developed in [12], and is capable of identify coherent co-clusters with irregular shapes. An overview of the algorithm is shown in Alg. 1. In the following, we explain each step in detail.

A. Co-clustering using tNMF

Given a DNS failure graph $\mathcal{G}$, as the first step in Alg. 1, we extract all the isolated components from $\mathcal{G}$. Though most of the components are fairly simple and small, there exist several large connected components which are comprised of loosely connected dense subgraphs, and thus are further decomposable. In the next step, we iteratively decompose each of these large components using the tri-nonnegative matrix factorization (tNMF) algorithm, which has been successfully applied to decompose (application) *traffic activity graphs* (TAGs) in [12]. In the following, we provide a brief overview of the tNMF algorithm in the context of decomposing DNS failure graphs.

Algorithm 1 Decomposing DNS failure graphs

```
Input: A DNS failure graph $\mathcal{G}$;
Obtain disconnected subgraphs $\mathbf{G} := \cup_i \mathcal{G}_i$;
for each $\mathcal{G}_i$ in $\mathbf{G}$ do
    Run tNMF to decompose $\mathcal{G}_i$ into $k \times l$ co-clusters;
    Filter noise in $\mathcal{G}_i$ by removing co-clusters with low densities;
    Merge dense co-clusters;
    Output all coherent co-clusters;
end for
```

Given a DNS failure graph $\mathcal{G}$ (or rather, a large connected component in $\mathcal{G}$) representing the interaction patterns of m hosts and n domain names (For simplicity, we abuse the notation by using $\mathcal{G}$ to represent a subgraph instead of the original DNS failure graph). Let $A_{m\times n}$ be the corresponding adjacency matrix of $\mathcal{G}$. The tNMF algorithm approximately *factorizes* $A_{m\times n}$ into three *low-rank nonnegative* matrices, $R_{m\times k}$, $H_{k\times l}$, and $C_{n\times l}$ so as to minimize the following objective function J, subject to the orthogonality constraints on R and C:

$$\min_{R\geq 0, C\geq 0, H\geq 0, R^TR=I, C^TC=I} J(R,H,C) = ||A - RHC^T||_F^2$$

where $||\cdot||_F$ is the Frobenius norm, and $k, l << \min(m, n)$. An algorithms is developed in [25] to solve this optimization problem by iteratively updating R, C and H.

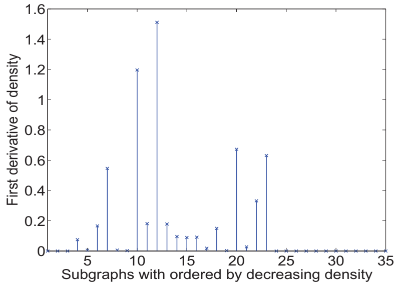


Fig. 4: Density change.

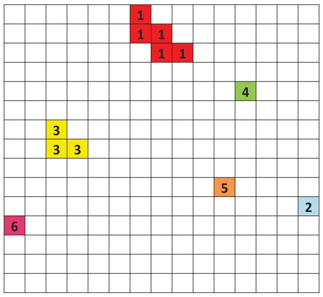


Fig. 5: Merging co-clusters.

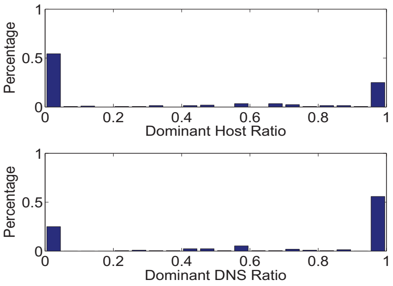


Fig. 6: Distributions of dhr and ddr.

In the context of our study, the decomposition results of the tNMF algorithm can be interpreted as follows. The matrices R and C divide the rows and columns into k host groups and l domain name groups, where $R_{\cdot p}$, $p = 1, \cdots, k$, and $C_{\cdot q}$, $q = 1, \cdots, l$, serve respectively as the "membership indicator" functions of the row groups and column groups. Assuming a *hard* co-clustering setting [12], we assign each host/domain name to only one row/column group with the largest entry in R/C (random assignment is used to break ties). We denote the new row and column membership indicator matrices in the hard co-clustering setting as $\hat{R}$ and $\hat{C}$, respectively.

One row group p and one column group q together form a subgraph or a co-cluster in $\mathcal{G}$ (we use subgraph and co-cluster interchangeably hereafter), and its density is computed as follows:

$$H_{pq} := \frac{(\hat{R}^T A \hat{C})_{pq}}{||\hat{R}_{\cdot p}||_1 \cdot ||\hat{C}_{q\cdot}||_1}, 1 \leq p \leq k,\ 1 \leq q \leq l, \qquad (1)$$

where $|| \cdot ||_1$ is the L_1-norm. The co-clusters with high H_{pq} (density) values correspond to dense subgraphs, while the ones with low H_{pq} values can be viewed as a loosely connected subgraphs with a small number of random links (or noisy edges). By filtering these weak connections or noisy edges, we can then extract the dense subgraphs from the DNS failure graph (or each of its large connected components).

B. Obtaining Coherent Co-clusters

The parameters k and l are two key parameters that determine the number of row groups and column groups, and therefore the total number of resultant co-clusters. Many approaches such as trial-&-error, model selection through statistical testing, and so forth, can be applied for selecting appropriate values for k and l. In this paper, we start with larger (likely than the "true") values for k and l (i.e., we first over-estimate k and l)[2], which yields *finer-grained* subgraphs or co-clusters. We then apply a *coherent* co-cluster selection process to merge these finer-fined subgraphs into more coherent subgraphs or co-clusters (with potentially "irregular" shapes). A similar approach has been applied in [26], which shows that such an approach is more effective in obtaining more coherent co-clusters than attempting to directly find the "true" values of k and l.

With such choices of k and l, we apply the tNMF algorithm to decompose a given DNS failure graph. We compute the densities for all the subgraphs H_{pq}'s thus extracted, and rank them in a decreasing order. We then use the change in the densities of subgraphs thus ranked to differentiate dense subgraphs from non-dense subgraphs, i.e., those that consist mainly of a few random, noisy edges. We use the graph in Fig. 2 as an example to illustrate how this is done, where we apply the tNMF method with $k = l = 15$. After ranking the subgraphs based on their densities, Fig. 4 shows the change in density of these subgraphs, where y-axis shows the relative change $(y_i - y_{i+1})/y_{i+1}$ of the (non-zero) density. We observe that the most significant change occurs between the 12th and the 13th subgraphs, and after the densities are much smaller after that.

After the noisy, non-dense subgraphs are removed, we can check to see whether some of the dense subgraphs can be merged to form more coherent co-clusters (with potentially irregular shapes). We merge two subgraphs if they share either a common host group or a common domain name group. Hence the co-clusters are formed by adjacent dense areas displayed in the density matrix H. Fig. 5 shows the merging results for the graph in Fig. 2: although after removing the noisy, non-dense subgraphs, we have obtained a total of 12 dense subgraphs; these 12 dense subgraphs essentially form 6 coherent co-clusters (after merging)– the numbers in Fig. 5 identify these 6 coherent co-clusters. Comparing to the other four co-clusters, co-cluster 1 and 3 do not have a typical box shape, thus they cannot be obtained with classical co-clustering algorithms (e.g., the standard tNMF algorithm in [12], which always produces box-(or rectangular) shaped co-clusters).

Until now, we can extract all the dense subgraphs (communities) from DNS failure graphs. In the next section, we analyze these subgraphs in detail and show that they are likely corresponding to different anomalous activities in the network.

V. Analysis of Co-clusters

After decomposition, the DNS failure graphs break into multiple coherent co-clusters (dense subgraphs). In this section, we provide a detailed analysis of the co-clusters extracted from our 3-month DNS trace.

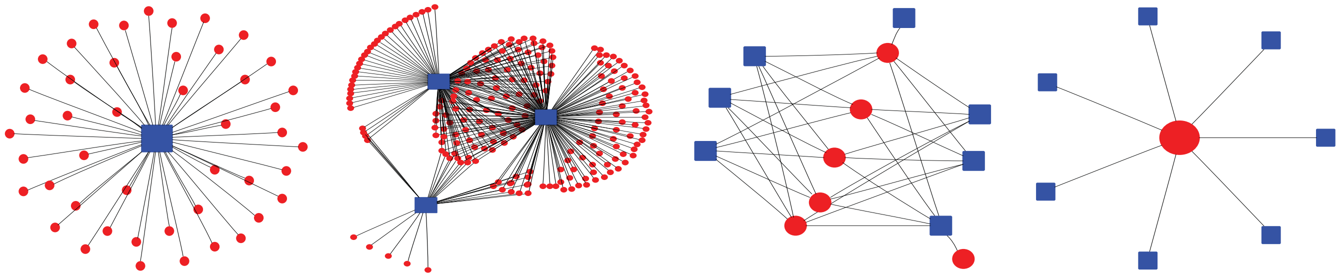
A. Categorizing Co-clusters

We categorize different co-cluster structures based on whether there are a few dominant hosts or a few dominant

[2] In our experiments, we choose $k = l = \lceil min(m, n)/30 \rceil$.

TABLE II: Categorization of identified co-clusters.

ID	Root cause	Pct.(%)	Details	Bi-mesh	Host-star	DNS-star
1	Trojan (Backdoor)	28.1	Variants of Dropper, Pakes!sd6, Rustock.E, Tidserv, WinFixer, Ertfor.A, Kraken, FakeAlert.a, Anti-Virus2008, Crypt.ta, etc.	63.2%	26.3%	10.5%
2	Spamming	25.2	Hosts querying for non-existing mail servers.	29.9%	70.1%	0
3	Domain-flux botnets	13.3	Conficker A/B, Torpig.	66.1%	33.9%	0
4	Peer-to-peer	5.2	Hosts querying for non-existing p2p servers.	100%	0	0
5	Unknown	28.1	Domain names not found in the data sources.	72.2%	20.1%	7.7%
	Total	100				



(a) Host-star (spamming)

(b) Bi-mesh (bot: Conficker A)

(c) Bi-mesh (trojan: Srizbi)

(d) DNS-star (spyware: Webhancer)

Fig. 7: Example of DNS failure subgraph structures.

domain names in the co-cluster. More specifically, let $A_{m\times n}$ denote the adjacency matrix corresponding to a particular co-cluster consisting m hosts and n domain names. Let $p_{i\cdot} := \sum_j a_{i\cdot} / \sum_{i,j} a_{i,j}$ and $p_{\cdot j} := \sum_i a_{\cdot j} / \sum_{i,j} a_{i,j}$ be the marginal probabilities of the rows and the columns, respectively. We define the *dominant host ratio* (dhr) as $dhr := -(\sum_i p_{i\cdot} \log p_{i\cdot}) / \log m$, which varies between 0 and 1. A dhr close to 0 implies there are a few dominant hosts that connect to far more domain names than other hosts in the same co-cluster; while a dhr close to 1 means all the hosts query approximately equal number of domain names. Similarly, we define the *dominant DNS ratio* (ddr) as $ddr := -(\sum_j p_{\cdot j} \log p_{\cdot j}) / \log n$ to identify dominant domain names. We say a co-cluster has a (likely) *host-star* structure if $dhr < \delta$ and $ddr > 1 - \delta$. In comparison, a (likely) *DNS-star* structure is defined if $dhr > 1 - \delta$ and $ddr < \delta$. If $dhr > \delta$ and $ddr > \delta$, we call such a structure a *bi-mesh*.

In Fig. 6, we show the distributions of dhr and ddr of all the co-clusters extracted from the daily DNS failure graphs in our dataset. We note that when a co-cluster is too small, we usually do not have enough evidence to interpret the meaning of that co-cluster. Meanwhile, the three structures are also less meaningful for small co-clusters. Therefore, we filter the co-clusters which contain less than 5 nodes (hosts plus domain names). Though the remaining co-clusters account for only 8% of all the co-clusters, they cover more than 42% hosts and 53% domain names. From the strong bi-modal shapes of both dhr and ddr distributions in Fig. 6, we choose $\delta = 0.1$ to separate the three types of structures.

We illustrate examples of the three different subgraph structures in Fig. 7[a-d], where blue boxes and red circles represent hosts and domain names, respectively. A link means a host has queried for the corresponding domain name. The host-star structure in Fig. 7[a] is due to a host querying for many non-existing domain names containing keywords like mail, mx, etc, most likely an instance of spamming activities. We show in Fig. 7[b] a bi-mesh structure caused by a set of domain-flux bots (Conficker A [9]). As we shall see in Section VI, this is because these bots access the domain name list from the same DGA algorithm. Another example of the bi-mesh structure is shown in Fig. 7[c], which corresponds to the activity of the trojan Srizbi [27]. An example of the DNS-star structure is displayed in Fig. 7[d], corresponding to 7 hosts querying for a non-existing domain name *webhancer.com*, which is related to a reported spyware activity [28].

B. Interpreting Co-clusters

Given the three types of co-cluster structures (or interaction patterns) between hosts and domain names, we next study the root causes of these different co-cluster structures. For each co-cluster, we first extract all the associated domain names. We then match these domain names against all the external data sources we have. For a matched domain name, we label it with the root cause specified by the data source. We then assign the co-cluster with the most dominant root cause. In Table II, we summarize all the co-clusters extracted from the daily DNS failure graphs using our dataset. Each row describes a specific category of co-clusters classified by the root cause. The second column shows the root cause of the co-cluster. The third column indicates the proportion of the co-clusters belonging to that category over all the observed co-clusters. We provide examples or explanation of each category in column 4. We further identify the percentages of co-clusters in each category that are bi-meshes, host-stars and DNS-stars (column 5-7).

From Table II, we observe that *trojan* (backdoor) is the most common root cause of the co-clusters, which accounts for 28.1% of the co-clusters in total. These detected trojan instances maintain a (usually hard-coded) list of domain names of the C&C servers where they can upload sniffed privacy information and download commands or updates. The domain names are associated with the C&C servers either through standard DNS registrar or using fast-flux mechanism [17]. The domain names hardly change after the trojans are released. Therefore, such domain names can be easily blocked or removed from the registrar once the trojan malware is detected. The failed DNS queries are caused by trojans querying the domain names that are already blocked or deleted from the DNS registrar. In general, these trojan instances contain a limited number of domain names, and hence the co-clusters in this category often exhibit bi-mesh (63.2%) or host-star (26.3%) structures. We note that although these trojan instances are detached from the C&C servers, they still remain as a threat since the specific exploits are not fixed on these hosts, therefore they are vulnerable to future attacks.

The second major root cause (25.2%) is the spamming activities. Hosts involved in such activities periodically query for a large number of non-existing mail servers, thereby showing dominant host-star structures. Most of these mail servers belong to large ISP networks and somehow have their domain names changed. We suspect the hosts are infected by certain worms or bots, which use a list of common mail servers. During certain time periods, these worms/bots become active and query for the mail server addresses to propagate spams. We also observe 29.9% of the co-clusters are bi-meshes, possibly due to different hosts equipped with the same email server list. No DNS-star is found in this category.

The third category is caused by domain-flux botnets. The bot master of a domain-flux botnet uses a domain name generation algorithm (DGA) to periodically create a new domain name list for the C&C servers and select a few of them to register. To avoid conflict with the existing registered domain names, the domain names from the DGA algorithm often consist of random-looking strings with either variable or fixed lengths. Every bot belonging to the same botnet is equipped with the same DGA to continually generate domain name list of the C&C servers. A bot tries to connect to the domain names in the list to reach the C&C servers. Since most domain names on the list are not registered, such bot activity often leads to a large number of (correlated) DNS query failures. For some of the domain-flux botnets, the DGA algorithms have been successfully reverse engineered [9], [11], [10]. We employ these reverse-engineered DGA algorithms to precompute the domain name list and use it to identify co-clusters caused by domain-flux bots. With this method, we find that in total 13.3% of all the co-clusters are due to domain-flux bots. Because the same bot instances utilize the same DGA algorithm, they hence show strong correlation. As a result, 86% of the co-clusters are bi-meshes, with another 14% are host-stars when only one bot instance from a particular domain-flux botnet is observed.

P2P activities contribute to 5.2% of all the co-clusters. The correlated DNS query failures happen when more than one hosts look up for the same p2p servers that no longer exist. All of the identified p2p activities are bi-meshes, accessing the same domain names, such as *66bt.cn* and *zingking.com*, etc.

The last category consists of 28.1% of all the co-clusters that we cannot identify their root causes based on the domain names. 72% of these co-clusters are bi-meshes, which we suspect are possibly caused by unreported anomalous activities. As we shall see in Section VI, we find a number of them may correspond to the activities of unreported domain-flux bots.

How are these dense subgraphs connected? Because the subgraphs represent heterogeneous suspicious activities and hence ideally they are isolated subgraphs in a DNS failure graph. However, by studying the removed weak links, we find that under several circumstances they will be connected to form large subgraphs. One possible reason is that a host may be infected by multiple malwares. For example, in Fig. 2, we find two hosts that are multiple infected, one of them is infected by both Conficker B and Horse, and the other is infected by Horse and Torpig. Another possible reason is that hosts infected by different malwares may share some other common behaviors. For example, we observe that P2P related DNS failures are likely to appear together with other infections, such as confickerB, Horse, Win32/Polip. As another example, hosts infected by different trojans may query the same (non-existing) mail servers for spamming purpose. In addition, a few edges are caused by hosts changing their dynamic IP addresses within the observation period.

VI. Evolution of DNS failure graphs

In this section, we explore the temporal properties of the DNS failure graphs. We first propose a best-effort linking algorithm to correlate co-clusters identified from daily DNS failure graphs on various days. We then differentiate subgraphs experiencing significant changes over time from the stable ones. At the end of the section, we show that many of the dynamic subgraphs are likely unreported domain-flux bots.

A. Tracking Co-cluster Changes

For a particular co-cluster, either hosts or domain names may change over time due to dynamic address allocation or the domain name generation schemes used by bots. In order to track the changes of co-clusters over time, we employ a best-effort approach which takes both factors into account.

Given a particular co-cluster $G_{i,t}$ from day t, let $\mathcal{H}_{i,t}$ and $\mathcal{D}_{i,t}$ be the sets of hosts and the domain names associated with $G_{i,t}$, respectively. We use the *Jaccard Similarity Coefficient (JSC)*[3] to measure the similarity between $G_{i,t}$ and every subgraphs $G_{j,t+1}$ from the following day $(t+1)$ to find the best match in terms of both the hosts and the domain names. In particular, we call $G_{j,t+1}$ the best match of $G_{i,t}$ if

$$j = argmax_j \max(JSC(\mathcal{H}_{i,t}, \mathcal{H}_{j,t+1}), JSC(\mathcal{D}_{i,t}, D_{j,t+1}))$$

and $\max(JSC(\mathcal{H}_{i,t}, \mathcal{H}_{j,t+1}), JSC(\mathcal{D}_{i,t}, \mathcal{D}_{j,t+1})) > \theta$. Fig. 8 shows the distribution of the JSCs for the best matches between the subgraphs on 01/05/2009 and those on 01/06/2009. Due to the bimodal shape, we choose $\theta = 0.6$ as the cut-off threshold in our experiments, i.e., a co-cluster has no best match if the maximum JSC value is less than 0.6. In this way,

[3] For two sets A and B, the JSC is defined as $|A \cap B|/|A \cup B|$.

we can track the changes of a particular subgraph by finding its best matches in the subsequent days recursively.

We use a simple criterion to differentiate stable co-clusters and dynamic ones based on the change of the domain names. We consider a co-cluster to be unstable over time if the maximum JSC between the domain name sets appearing at the first day and any of the subsequent days is less than 0.1[4]. In addition, we only focus on the co-clusters that last for more than one week. For co-clusters with a shorter life, we need more observations to study their changes.

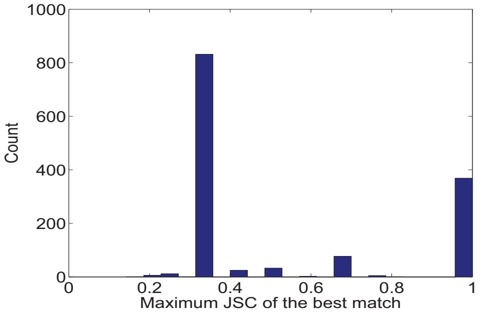


Fig. 8: Maximum JSC.

There are totally 20 co-clusters that last for more than 1 week, where 12 of them are stable co-clusters. Not surprisingly, these stable co-clusters are caused by correlated trojan malware activities, which access the same set of domain names all the time. However, we identify 8 co-clusters that are more dynamic (i.e., with significant domain name changes). We next study and interpret these dynamic co-clusters.

B. Analyzing Dynamic Co-clusters

For the 8 co-clusters with significant domain name changes, using the reverse-engineered DGA algorithm, we find that four co-clusters are related to three types of domain-flux bots: Conficker A, Conficker B and Torpig (the Conficker B bots form two separate co-clusters, due to one particular day when no bot instance sends out DNS queries). In fact, these 4 co-clusters cover all the domain-flux bots belonging to these three botnets without any false alarm. In other words, our method can identify these three types of bots with 100% accuracy purely by exploring the correlation in DNS failure graphs.

In addition, the remaining 4 co-clusters are labeled as *unknown* and cover 53.2% of the unknown domain names. These co-clusters demonstrate similar patterns as those of the reported domain-flux bots. We next provide a detailed analysis of these co-clusters to show that they are also likely corresponding to unreported domain-flux bots.

We start by examining at the patterns in the domain names. Table III shows some examples of domain names from these 4 candidates. Candidate A uses *.com*, *.net* or .cc for the top level domain name while the other three candidates only use *.com*. The second level domain names from these 4 candidates are apparently random strings of a variable length (candidate A) or a fixed length (B of length 8, C of length 10 and D of length 20). This indicates that these domain names are likely generated by machines (using certain algorithms) other than by human beings.

TABLE III: Domain name patterns.

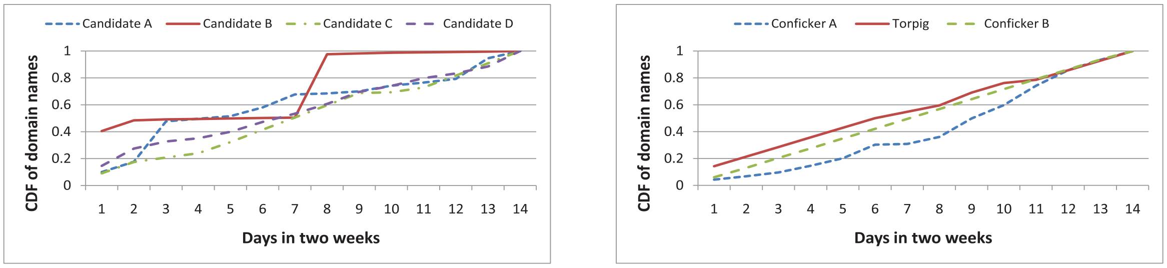
Candidate A	Candidate B	Candidate C
gkymopkcffqt.com ymtyupvty.net fqhfaia.cc tbllutksqg.com	guxwivkb.com sbttwbkh.com xbhsxdgk.com svvwdddw.com	aufutmguua.com ncamnsdtxa.com hlhxeezzsd.com lpqrmgiwln.com
Candidate D		
guyyruldrbrbqyfxdtnb.com, dlqrhudtjiajuopbagwg.com hqcwbspyvdpmhrejvhdi.com, wvkafndfedfoxkcdlimw.com		

We next study the cycles of domain name changes of the 4 candidates. Fig. 9[a] shows different lengths of cycles of these 4 candidates, where the x-axis represent the number of days (relative to the time when the bot instances begin to be observed) and the y-axis stands for the cumulative number of unique domain names appearing over time. We observe that except for the candidate B which has a cycle length of 1 week, all the others have a cycle length of 1 day. In comparison, we show the cycles of the three known bots in Fig. 9[b]. All the three known bots have a cycle length of 1 day.

At the end of the two-week period, the total number of unique domain names observed for each candidate also varies significantly compared with the known bots. For example, Torpig bots only have 42 unique domain names after 2 weeks (3 new domain names generated by the DGA per day). In contrast, candidate C has more than 42K in 2 weeks, where around 3K new domain names are observed per day. To further differentiate whether 4 candidates are the variants of the known bots, we compare the hosts associated with each of them. In fact, there is no IP address shared by the candidates and the known bot instances, suggesting these candidates are plausibly unreported domain-flux bots.

Unlike the large number of failed domain names, the registered domain names and the successful queries are of special interest to us, because they provide hints on the IP addresses of the C&C servers as well as the botmasters (who registered these domain names). For all these 4 candidates, we extract the associated domain names with similar patterns (as identified in Table III) from *both successful and failed DNS queries*. We find there is no successful domain name query for candidate A, C and D, possibly due to the short observation time period and the small sample size. We do observe 1 IP address returned for candidate B, which is registered 7 days before the first access toward this address. However, since the host may be infected by multiple malwares, we need further evidence to verify that this address is indeed a C&C server address for a domain-flux botnet. An interesting observation for candidate B is that a few of the failed domain names are indeed registered. For example, *xnihxzatff.com* and *sjfnannvwv.com* are registered on 01/06/2009. However, the hosts are observed to access them only on 01/01/2009, which results in failed DNS queries. We suspect that this may be

[4] We note that the threshold 0.1 is set to address the cases of domain-flux bots with different domain name generation cycles.



(a) Domain-flux bot candidates

(b) Known domain-flux bots.

Fig. 9: Identifying cycles of domain name changes.

caused by either the synchronization problem between the registration process and the DGA algorithm, or the DGA may generate domain names that may repeat in future.

In summary, even though the use of DGA algorithms to generate domain name lists and query accordingly is a common characterization of domain-flux bots, the query patterns in term of number of domain names and frequency may vary for different kinds of domain-flux bots. Existing studies detect domain-flux bots by identifying a significant increase in DNS query failures [3]. Such methods may miss the domain-flux bots with less intensive activities, such as the Torpig bots [10], which only generate 3 domain names per day. Reverse engineering based methods (e.g., [9], [11]) have a much higher accuracy, but are more expensive. In contrast, by correlating DNS activities among hosts, our method can detect domain-flux accurately without the need to access to extra traffic information or knowledge about individual bots.

VII. Conclusion

In this paper, we proposed an approach for identifying and classifying network anomalies based on unproductive DNS traffic. We advanced the notion of DNS failure graphs to capture the interaction between hosts and failed domain names. We then applied a statistical tri-nonnegative matrix factorization technique for extracting coherent co-clusters (dense subgraphs) from DNS failure graphs. Analysis on a 3-month DNS trace captured at a large campus network indicated most of such co-clusters correspond to a variety of network anomalies which often exhibit different subgraph structures. Temporal analysis on these co-clusters identified 8 persistent co-clusters representing groups of hosts collectively query for different sets of domain names over time. Four of them belong to known domain-flux bots; while the remaining four co-clusters are plausibly due to unreported domain-flux bots.

Acknowledgement

The work is supported in part by the NSF grants CNS-0626812, CNS-0905037 and CNS-1017647, and the DTRA Grant HDTRA1-09-1-0050. Part of the work was done during a summer internship by the first author at Bell Labs.

References

[1] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of configuration errors on DNS robustness," in *SIGCOMM'04*, 2004.

[2] D. Dagon, "Botnet detection and response, the network is the infection," in *OARC workshop*, 2005.

[3] Z. Zhu, V. Yegneswaran, and Y. Chen, "Using failure information analysis to detect enterprise zombies," in *SecureComm'09*, 2009.

[4] D. Plonka and P. Barford, "Context-aware clustering of DNS query traffic," in *IMC'08*, 2008.

[5] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *LEET'08*, 2008.

[6] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," in *SIGCOMM '08*, 2008.

[7] J. John, A. Moshchuk, S. Gribble, and A. Krishnamurthy, "Studying spamming botnets using botlab," in *NSDI'09*, 2009.

[8] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum, "Botgraph: large scale spamming botnet detection," in *NSDI'09*, 2009.

[9] P. Porras, H. Saidi, and V. Yegneswaran, "Conficker C analysis," http://mtc.sri.com/Conficker/addendumC/.

[10] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *CCS'09*, 2009.

[11] "Conficker working group," http://www.confickerworkinggroup.org/wiki/.

[12] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, "Unveiling core network-wide communication patterns through application traffic activity graph decomposition," in *SIGMETRICS '09*, 2009.

[13] K. Sato, K. Ishibashi, T. Toyono, and N. Miyake, "Extending black domain name list by using co-occurrence relation," in *LEET'10*, 2010.

[14] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *IMC '04*, 2004.

[15] "The honeynet project," http://www.honeynet.org/.

[16] Y. Jin, Z.-L. Zhang, K. Xu, F. Cao, and S. Sahu, "Identifying and tracking suspicious activities through ip gray space analysis," in *MineNet '07*, 2007.

[17] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and detecting fast-flux service networks," in *NDSS'09*, 2009.

[18] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Real-time detection of fast flux service networks," *Conference For Homeland Security, Cybersecurity Applications and Technology*, 2009.

[19] "MX Toolbox Blacklists," http://www.mxtoolbox.com/blacklists.aspx.

[20] "ThreatExpert Report," http://www.threatexpert.com.

[21] P. Porras, H. Saidi, and V. Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points," http://mtc.sri.com/Conficker/.

[22] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci, "Unconstrained Endpoint Profiling (Googling the Internet)," in *SIGCOMM '08*, 2008.

[23] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are ip addresses?" in *SIGCOMM '07*, 2007.

[24] "Graphviz - graph visualization software," http://www.graphviz.org/.

[25] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *KDD'06*, 2006.

[26] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. Dhillon, "A scalable framework for discovering coherent co-clusters in noisy data," in *ICML'09*, 2009.

[27] "Srizbi," http://www.threatexpert.com/threats/trojan-srizbi-sd6.html.

[28] "Webhancer," http://www.spywareguide.com/spydet_26_webhancer.html.