Identifying Suspicious Activities through DNS Failure Graph Analysis

Nan Jiang*, Jin Cao†, Yu Jin*, Zhi-Li Zhang*, Li Erran Li†
*Computer Science Dept., University of Minnesota †Bell Laboratories, Alcatel-Lucent

ABSTRACT

As a key approach to securing large networks, existing anomaly detection techniques focus primarily on network traffic data – the sheer volume of such data often render detailed analysis very expensive and reduce 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* – which captures the interactions between hosts and failed domain names. We apply a tri-nonnegative matrix factorization technique to recursively 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., which often exhibit distinguishable subgraph structures. In addition, by exploring the temporal properties of the co-clusters, we have identified new anomalies that likely correspond to unreported domain-flux bots.

1. 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 at the time of the query. Such a failure may be caused by a mis-typed host name or URL by a human user, and occasionally by DNS misconfigurations by human operators. However, as pointed out in several recent studies [1, 2, 3], a large portion of DNS query failures can be attributed to other causes, e.g., (see Section 2 for an analysis and classification of DNS query failures). In particular, as shown in [2], 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 [4, 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 DNS names of failed DNS queries and hosts issuing such queries. 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 with the same malware (e.g., belonging to the same botnet) usually query for the same, similar or otherwise correlated set of DNS names, for instance, to locate the Command&Control (C&C) servers, malware hosting sites, stolen data storage servers, etc. To evade detection, the DNS names used by these malicious activities often change frequently (i.e., in domain-flux); those that do not flux frequently often are blacklisted and blocked after detection. Hence queries for these DNS names frequently result in *correlated* failures, which manifest themselves as a *dense* subgraph in the DNS failure graph. Such dense subgraphs therefore capture the strong *interaction patterns* between a set of hosts and a set of DNS 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 DNS names in a DNS failure graph?*

To answer this questions, 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 DNS names. While these giant components are fully 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 DNS names – as representing and corresponding to a single type of anomaly. We therefore apply a (statistical) graph decomposition technique based on the *tri-nonnegative matrix factorization (tNMF)* [9] to recursively decompose a DNS failure graph and extract dense (bipartite) subgraphs, or *co-clusters*, containing 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 host dominates by sending a large number of DNS queries; 2) a *DNS-star*, where a domain name attracts 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 DNS names. In comparison, the host-stars are primarily the artifacts of spamming activities involving queries for expired DNS 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 DNS 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 analyzing 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 are likely bots or Trojan instances with a list of hard-coded DNS names for querying C&C and other servers. In contrast, we also find that several co-clusters are associated with a set of DNS names that flux over time. Analyzing the patterns of DNS 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 DNS 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 Related Work. 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 demonstrate how the tNMF graph decomposition method can be applied to extract dense subgraphs or co-clusters of hosts and DNS names that exhibit strong and coherent interaction patterns; 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. As mentioned earlier, our work is motivated by earlier works such as [1] which first points out using DNS queries for detecting bots, [2] 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 [3] 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 demonstrate how it can be used to identify and differentiate suspicious activities using *correlation of 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. In carrying out this study, we have also freely leveraged the results and insights from studies of domain-flux, spam, p2p and other botnets (say, e.g., [8, 10, 11, 4, 5, 6, 7]).

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. In Section 2, we analyze the failed DNS queries and introduce the notion of DNS failure graphs. We then propose a tNMF based algorithm for decomposing DNS failure graphs into strongly connected subgraphs in Section 3. Section 4 presents the classification and interpretation of these dense subgraphs and their temporal properties are studied in Section 5. Finally, Section 6 concludes the paper.

2. DNS TRAFFIC AND 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

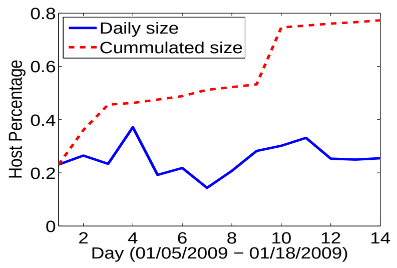

Figure 1: Size of the largest DNS failure graph over time.

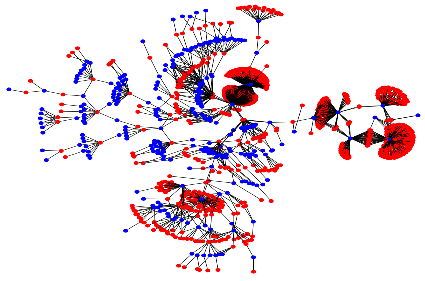

Figure 2: The largest DNS failure subgraph from 01/05/2009.

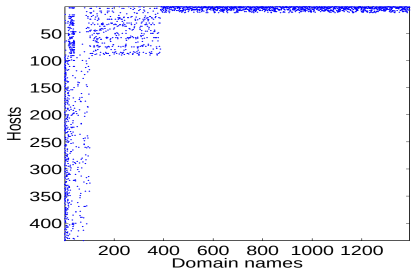

Figure 3: Block structures after row and column rotations.

collected DNS data contains DNS requests and responses from all hosts within the campus network for resources located 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 the 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 DNS 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.

2.1 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 1 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" [3], where several anti-spam and anti-virus services employ DNS to notify a querying host whether the requested DNS 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. 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 are temporarily unresolvable. DNS misconfigurations such as a query for *www.example.com.example.com* (such "recursive DNS names" are likely due to Window default DNS suffix configured at client machines) account for 7.87% of the all failed DNS queries, while DNS typos, which are likely caused by users mistyping a few alphabetics of the desired domain name, account for 2.26%.

For the remaining failed DNS queries, we look up the target DNS name in each failed query in a number of auxiliary data sources, including various blacklists [12], security logs [13], botnet related domain names obtained via reverse engineering [14], and information obtained by googling the

Type	Pct (%)	Examples
DNS Overloading	32.37	spamcop.net, surbl.org
Server errors	28.01	unresolvable domain names in a server farm
Misconfigurations	7.87	www.example.com.example.com
Typos	2.26	googloe.com, encyclopiedea.net
Known Threats	2.08	g43gwef.com, antispyware2008xp.com
P2P	0.75	66bt.cn, zingking.com
Unknown	27.33	vuuewgkt.com, dehpydjsi.cn

Table 1: Categories of failed DNS queries.

Internet [15]. If a target DNS 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 DNS names are associated with p2p applications and services found on-line. Finally, we cannot properly attribute the 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*. Most of the targeted DNS names contain random-looking strings, and as we shall see later, most of them are likely associated with suspicious anomalies, e.g., domain-flux botnet activities.

2.2 DNS Failure Graphs and Properties

The aforementioned method of identifying potential threats in "unproductive" DNS traffic by matching the target DNS names in failed DNS queries against data sources of known security threats is rather time-consuming; its 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 1, a significant portion (27%) of failed DNS queries cannot be attributed to *known* threats. The large majority of these failed DNS queries contain DNS 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.

Before we perform the DNS failure graph analysis, we first "cleanse" the DNS failed queries by filtering[1] failed

[1]We have developed a heuristic cleansing procedure to automati-

DNS queries that are attributable to "normal" network activities such as DNS overloading, server errors, misconfigurations and typos. 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[2].

We now formally define *DNS failure graphs*: Given an observation period T, 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) DNS 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 an host $h \in \mathcal{H}$ and a DNS 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[3] 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 DNS names each day. Each daily DNS failure graph is often composed of 1000 or more *isolated* components (subgraphs): each component is fully connected, but there are no edges connecting any two (connected) components (i.e., the components are isolated from each other). Despite the large number of these isolated components – a large majority of them are small, there exists 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) get connected by one 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 are 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 Graphviz [16], as shown in Fig. 2, where the blue nodes and red nodes represent hosts and DNS 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 DNS names they query; in other words, there are *strong interaction patterns* that connect the set of hosts and the set of DNS 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 DNS 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 DNS names, other contain a large number of hosts but a smaller number of DNS names, and yet other contain both relatively large numbers of hosts and DNS names. These different interaction patterns between the hosts and DNS names suggest that the hosts involved are likely engaging in different kinds of suspicious activities. These visual analyses suggest that the largest connected components can be further *decomposed* into dense subgraphs, which more likely correspond to correlated behaviors. In the next section, we present a general and effective methodology for automatically extracting these dense subgraphs or "communities".

3. DECOMPOSING DNS FAILURE GRAPHS

We present an algorithm for decomposing, and extracting dense subgraphs from, DNS failure graphs. This algorithm is based on the tNMF-based graph decomposition technique developed in [9]. An overview of the algorithm is presented in Alg. 1. In the following, we explain each step in detail.

3.1 Co-clustering using tNMF

Given a DNS failure graph $\mathcal{G}$, as the first step in Alg. 1,

cally filter these "normal" DNS query failures. For example, we filter overloaded DNS query failures by matching the responders of failed DNS queries against a list of known anti-spam/anti-malware sites, and adopt a similar approach as proposed in [1] 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.

[2]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 technique.

[3]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 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.

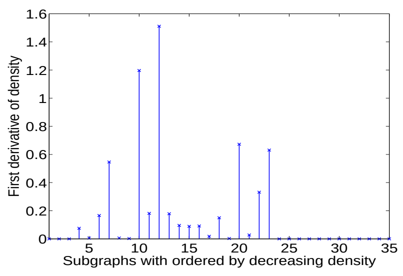

Figure 4: Density change.

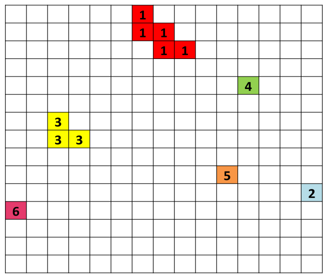

Figure 5: Merging co-clusters.

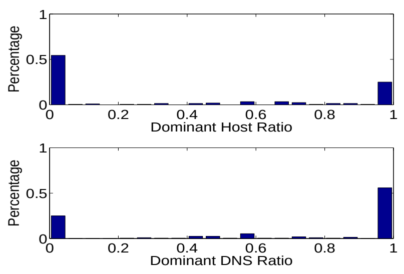

Figure 6: Distributions of *dhr* and *ddr*.

Algorithm 1 Decomposing DNS failure graphs

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

we extract all the isolated components from $\mathcal{G}$. 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 [9]. In the following, we provide a brief overview of the tNMF algorithm in the context of decomposing DNS failure graphs.

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 DNS names. Let $A_{m\times n}$ be the corresponding adjacency matrix of $\mathcal{G}$. The tNMF algorithm approximately *factorize* $A_{m\times n}$ 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:

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

where $||\cdot||_F$ is the Frobenius norm, and $k, l << \min(m,n)$. An iterative algorithms is developed in [17] to solve this optimization problem.

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 DNS 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 [9], we assign each host/DNS 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 form a subgraph in $\mathcal{G}$, and its density H_{pq} is computed as follows:

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

and $||\cdot||_1$ is the L_1-norm. The subgraphs 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).

3.2 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 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)[4], 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 [18], 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 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 graphs, 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 relative change (defined as $(H_i - H_{i+1}/H_i)$) of the (non-zero) densities. We observe that the most significant change happens when the number of dense subgraphs is chosen to be 12. Because of their small densities, the remaining subgraphs are

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

considered as containing mainly random, noisy edges and thus discarded.

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. 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 four other co-clusters, co-cluster 1 and 3 do not have a typical box shape, thus they cannot be obtained when using a classical co-clustering algorithm (e.g., the standard tNMF algorithm in [9]), which always produces box-(or rectangular) shaped co-clusters.

4. 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.

4.1 Characterizing Co-Clusters

We categorize different co-cluster structures based on whether there is a dominant host or a dominant domain name 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 is a dominant host that connects to far more domain names than other hosts in the same co-cluster. 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 daily DNS failure graphs over a 3-month time period. We note when a co-cluster is too small, we usually do not have enough evidence to interpret the meaning of that co-cluster. In addition, these three structures are also less meaningful for small co-clusters. Therefore, we only analyze the co-clusters which contain at least 5 nodes (hosts plus domain names). Though such co-clusters account for only 8% of all the co-clusters, they cover more than 42% hosts and 53% domain names. From the strong bimodal shapes of both dhr and ddr distributions in Fig. 6, we choose $\delta = 0.1$ to distinguish the three types of structures.

4.2 Interpreting Co-clusters

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 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 a co-cluster with the most dominant root cause. In Table 2, we summarize all the co-clusters extracted from the daily DNS failure graphs. 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 from that category out of 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 are bi-meshes, host-stars and dns-stars (column 5-7).

From Table 2, we observe that Trojan (backdoor) is the most common root cause, which accounts for 28.1% co-clusters in total. These detected Trojan instances maintain a hard-coded list of domain names of the C&C servers where they can upload sniffed privacy information and download commands or updates. These domain names hardly change after the Trojans are released. Therefore, such domain names can be easily banned or removed from the registrar once the Trojans are detected and thereby resulting in DNS query failures when a Trojan instance queries for these blocked domain names. The co-clusters in this category often exhibit bi-mesh or host-star structures. We note that although these Trojan instances are detached from the C&C servers, the associated hosts are still vulnerable to future attacks since the exploits are not yet fixed.

The second major root cause (25.2%) is the spamming activity. Hosts involved in spamming activities periodically query for a large number of mail servers, where many of these mail servers belong to large ISP networks and somehow have their domain names changed. Therefore, these co-clusters are dominantly host-stars. 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. A bot belonging to the botnet is equipped with the same DGA and keeps refreshing the domain name list of the servers. The bot then tries to connect to the domain names in the list to reach the C&C server. 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 [8, 10, 11]. We employ these reverse-engineered DGA algorithms to precompute the domain name list and use it to identify co-clusters caused by

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	
5	Unknown	28.1	Domain names not found in the data sources.	72.2%	20.1%	7.7%
	Total	100				

Table 2: Categorization of identified co-clusters.

domain-flux bots. With this method, we find totally 13.3% of all the co-clusters are due to domain-flux bots. In this category, 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 failed domain names *66bt.cn* and *zingking.com*.

The last category consists of 28.1% of all the co-clusters that we cannot find their root causes based on the domain names. We suspect that these co-clusters are likely caused by unreported anomalous activities. As we shall see in Sec. 5, we find that a number of unknown co-clusters behave similarly as the co-clusters caused by known domain-flux bots.

Discussion on the removed weak links. Because the subgraphs represent heterogeneous suspicious activities and hence ideally they are isolated subgraphs in the DNS failure graphs. However, by studying the removed weak links, we find that under several circumstances they will be connected to form large subgraphs. First, many of them are connected due to one single host participating in two activities, possibly due to multiple infection and dynamic IP allocation. For example, we find hosts that are infected by both Conficker A/B and the Trojan Horse. Second, hosts in different subgraphs may access other failed domain names, corresponding to email servers or p2p servers.

5. 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 at different times. We then differentiate subgraphs experiencing significant changes over time from the stable ones. At the end of the section, we show that the unstable subgraphs are likely unreported domain-flux bots.

5.1 Tracking Co-cluster Changes

For a particular co-cluster, both hosts and domain names may change over time due to dynamic address allocation and certain domain name generation schemes. 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) [5] 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. 7 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 JSC value is less than 0.6. In this way, 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 variable ones based on the change of the domain names. We consider a co-cluster to be stable over time if the JSC between the domain name sets appearing at the first day and the last day is less than 0.1[6]. In addition, we only focus on the co-clusters that last for more than 1 week. For co-clusters with a shorter life, we need more observations to study their changes.

5.2 Analyzing Potential Domain-Flux Bots

While most stable co-clusters (more than 90%) correspond to the activities of different Trojan instances, we identify 8 co-clusters with significant domain name changes. Using the reverse-engineered DGA algorithm, we find 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 addition, the remaining 4 co-clusters also 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 looking at the patterns in the domain names. Table 3 shows the 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

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

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

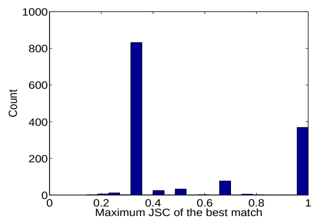

Figure 7: Maximum JSC.

Candidate *A*	Candidate *B*	Candidate *C*
ymtyupvty.net	sbttwbkh.com	ncamnsdtxa.com
fqhfaia.cc	xbhsxdgk.com	hlhxeezzsd.com
ppewqutd.com	wsisjxde.com	ywtfpxtwop.com
Candidate *D*		
guyyruldrbrbqyfxdtnb.com		
dlqrhudtjiajuopbagwg.com		
hqcwbspyvdpmhrejvhdi.com		

Table 3: Domain name patterns.

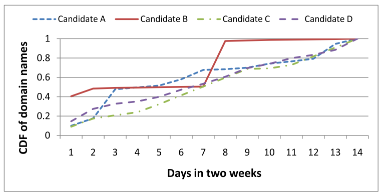

Figure 8: Identifying cycles of domain name changes.

.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 these domain names are likely to be generated by machines (following some algorithms) other than human beings.

We next study the cycles of domain name changes of the 4 candidates. Fig. 8 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. Similar as the three known domain-flux bots, candidate *A*,*C* and *D* also have a one-day cycle. In comparison, candidate B has a cycle of 1 week. 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 comparison, 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, indicating these candidates are likely unreported domain-flux bots.

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, respectively. However, the hosts are observed to access them only on 01/01/2009, which results in failed DNS queries. We suspect this may be 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.

6. 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 indicates most of such co-clusters correspond to a variety of network anomalies that often exhibit different subgraph structures. Temporal analysis on these co-clusters identifies 4 co-clusters plausibly due to unreported domain-flux bots.

7. REFERENCES

[1] D. Dagon, "Botnet detection and response, the network is the infection," in *OARC workshop*, 2005.
[2] Z. Zhu, V. Yegneswaran, and Y. Chen, "Using failure information analysis to detect enterprise zombies," in *SecureComm'09*, Athens, Greece, 2009.
[3] D. Plonka and P. Barford, "Context-aware clustering of dns query traffic," in *IMC'08*, Vouliagmeni, Greece.
[4] 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.
[5] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," in *SIGCOMM '08*, 2008.
[6] J. John, A. Moshchuk, S. Gribble, and A. Krishnamurthy, "Studying spamming botnets using botlab," in *NSDI'09*, 2009.
[7] 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.
[8] P. Porras, H. Saidi, and V. Yegneswaran, "Conficker C analysis," http://mtc.sri.com/Conficker/addendumC/.
[9] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, "Unveiling core network-wide communication patterns through application traffic activity graph decomposition," in *SIGMETRICS '09*, 2009.
[10] "Conficker working group," http://www.confickerworkinggroup.org/wiki/.
[11] 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.
[12] "MX Toolbox Blacklists," http://www.mxtoolbox.com/blacklists.aspx.
[13] "ThreatExpert Report," http://www.threatexpert.com.
[14] P. Porras, H. Saidi, and V. Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points," http://mtc.sri.com/Conficker/.
[15] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci, "Unconstrained Endpoint Profiling (Googling the Internet)," in *Proc. of ACM SIGCOMM '08*, 2008.
[16] "Graphviz - graph visualization software," http://www.graphviz.org/.
[17] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *Proc. of ACM KDD*, 2006.
[18] 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.