A Statistical Approach to Detecting Low-Throughput Exfiltration through the Domain Name System Protocol

A Statistical Approach to Detecting Low-Throughput Exfiltration through the Domain Name System Protocol

Emily Joback
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
Emily.Joback@ll.mit.edu

Steven R. Gomez
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
Steven.Gomez@ll.mit.edu

Leslie Shing
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
Leslie.Shing@ll.mit.edu

Steven Jorgensen
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
Steven.Jorgensen@ll.mit.edu

Kenneth Alperin
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
Kenneth.Alperin@ll.mit.edu

Gabe Elkin
Lincoln Laboratory Massachusetts Institute of Technology Lexington, MA, USA
GabeE@ll.mit.edu

ABSTRACT

The Domain Name System (DNS) is a critical network protocol that resolves human-readable domain names to IP addresses. Because it is an essential component necessary for the Internet to function, DNS traffic is typically allowed to bypass firewalls and other security services. Additionally, this protocol was not designed for the purpose of data transfer, so is not as heavily monitored as other protocols. These reasons make the protocol an ideal tool for covert data exfiltration by a malicious actor. A typical company or organization has network traffic containing tens to hundreds of thousands of DNS queries a day. It is impossible for an analyst to sift through such a vast dataset and investigate every domain to ensure its legitimacy. An attacker can use this as an advantage to hide traces of malicious activity within a small percentage of total traffic. Recent research in this field has focused on applying supervised machine learning (ML) or one-class classifier techniques to build a predictive model to determine if a DNS domain query is used for exfiltration purposes; however, these models require labelled datasets. In the supervised approach, models require both legitimate and malicious data samples, but it is difficult to train these models since realistic network datasets containing known DNS exploits are rarely made public. Instead, prior studies used synthetic curated datasets, but this has the potential to introduce bias. In addition, some studies have suggested that ML algorithms do not perform as well in situations where the ratio between the two classes of data is significant, as is the case for DNS exfiltration datasets. In the one-class classifier approach, these models require a dataset known to be void of exfiltration data. Our model aims to circumvent these issues by identifying cases of DNS exfiltration within a network, without requiring a labelled or curated dataset. Our approach eliminates the need for a network analyst to sift through a high volume of DNS queries, by automatically detecting traffic indicative of exfiltration.

CCS CONCEPTS

• **Security and privacy**; • **Network security**; • **Security protocols**; • **Networks**; • **Network protocols**; • **Application layer protocols**; • **Computing methodologies**; • **Machine learning**; • **Learning paradigms**; • **Unsupervised learning**;

KEYWORDS

DNS, Exfiltration, Clustering, Outlier Detection, Data Mining

ACM Reference Format:
Emily Joback, Steven R. Gomez, Leslie Shing, Steven Jorgensen, Kenneth Alperin, and Gabe Elkin. 2020. A Statistical Approach to Detecting Low-Throughput Exfiltration through the Domain Name System Protocol. In *2020 Workshop in DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security (DYNAMICS 2020), December 07, 2020, Lexington, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3477997.3478007

1 INTRODUCTION

The Domain Name System (DNS) is a hierarchical distributed database that resolves human-readable domain names to an Internet Protocol (IP) address. This protocol is an integral component of the Internet, and the functional driver behind web browsing and a large proportion of general Internet activities. Because reliability and timeliness of this service is critical, there are limited security restrictions on DNS (e.g., DNS traffic firewall exceptions [1]). Additionally, traffic over DNS is not monitored to the same degree as other services that exist explicitly for data transfer (e.g. File Transfer Protocol (FTP)) [2]. The lack of substantial security and minimal monitoring of the protocol make it an ideal tool for covert data exfiltration by a malicious actor.

Data exfiltration over DNS usually requires DNS tunneling methods as a means to establish a discrete connection between a system within a network and an external host controlled by the malicious actor. A tunnel is created when the malicious actor compromises a computer within a network, and resolves the domain name to the IP address of a machine controlled by the attacker.

DYNAMICS 2020, December 07, 2020, Lexington, MA, USA

ACM ISBN 978-1-4503-8714-9/20/12.
https://doi.org/10.1145/3477997.3478007

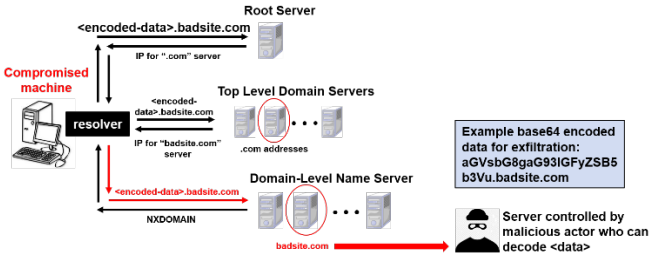

Figure 1: Using the DNS protocol for exfiltration purposes

DNS exfiltration occurs when this protocol is abused to export data off an internal host machine. Legitimate cases of DNS exfiltration do exist. For example, security services such as McAfee use the DNS protocol to offload information pertinent to their tool [3]. However, this same method can be utilized to exfiltrate data for malicious intent. One well known example is the case where information on 56 million credit and debit cards was stolen from customers of Home Depot; a similar incident occurred with Sally Beauty [4, 5]. Overall, DNS has been increasingly used as a vector to extract sensitive information [6].

A DNS query, called a Fully Qualified Domain Name (FQDN), consists of substrings separated by dots, called labels. Queries can contain up to 127 labels, where each label has at most 63 characters, and the total length of the domain string does not exceed 253 characters. The top level domain is the part of the domain string following the last dot (e.g. “com”, “edu”, and “gov”). The second level domain includes the substring preceding the top level domain (e.g. “google.com”, “mit.edu”, and “nasa.gov”). The subdomain is the remainder of the DNS query string which precedes the second and top level domain. Two methods of DNS lookups exist to resolve the IP address of a FQDN: recursive and iterative. When a domain name is queried by the DNS client, the request is first sent through a resolver to a root server. Root servers exist throughout the world, and each contain information on top level domains. In the recursive method, the request is passed from the root server to the appropriate top level domain server on behalf of the DNS client, and from there to the domain level name server. The domain level name server that knows the IP address for the requested domain will send this information back to the original host. In the iterative method, the requests pass through the same set of servers, but at each step the name of the next server is sent back to the resolver. It is then the responsibility of the resolver to iteratively work through this process with each of the servers until an IP address is retrieved. For low-throughput exfiltration, such as the Home Depot attack, the data is exfiltrated in small chunks at a time, by encoding the data chunks and placing them within the subdomain of a DNS query. An example of this attack is illustrated in Figure 1.

A typical company or organization’s network can contain traffic containing between tens to hundreds of thousands of unique domains daily. Inspecting each individual domain for signs of exfiltration would be an impossible task for a network analyst. Therefore a tool is needed to automatically filter out the majority of legitimate traffic, and retain only the few domains that are most likely malicious for further investigation by the network analyst.

2 PREVIOUS WORK

Abuse of the DNS protocol first became an evident issue in the early 2000s with the introduction of the Linux based tool Nameserver Transfer Protocol (NSTX), which made it possible for IP packets to be tunneled within DNS queries [7]. Following this, additional tools made their appearance [8] (e.g.: DNSCat [9], Iodine [10], TUNS [11], Dns2TCP [12], and OzyManDNS [13]). In the two decades since the introduction of NSTX, the transmission of data over DNS has been studied extensively. The research conducted in this arena can be divided into two categories: (1) traffic analysis and (2) payload analysis [14]. Traffic analysis looks at aggregate statistics across the data over time, such as the number of DNS requests to a domain/IP address or the number of received NXDomain responses [15-17]. Payload analysis investigates the data encoded within the subdomain of a DNS query, such as length or character frequency analysis of the query [15, 18]. Earlier studies focused on detecting specific tunneling tools, but more recently the focus has shifted to the detection of any DNS tunnel, using supervised machine learning methods [19-22]. Studies focused on DNS tunneling have been primarily concerned with the use of establishing a tunnel for command and control purposes, or for larger data exfiltration using the TXT record of the DNS query. These methods are tuned for larger exfiltration attempts and generic tunneling, and do not focus on low-throughput exfiltration which is less easily detected.

Low-throughput exfiltration is an attack where data is encoded and placed within the subdomain of DNS queries, in small chunks intermittently over time. Researchers have investigated methods involving either supervised machine learning techniques or one-class classifiers towards detecting this type of exfiltration. Research described in [23-24] represents recent work regarding implementation of machine learning techniques. The issue with these supervised machine learning methods is the lack of available labelled datasets containing both benign and malicious DNS traffic required to train the models. To circumvent this issue, studies have looked to synthetically produce an artificial dataset for training and testing. However, doing this can introduce inherent bias into the model, resulting in a solution that does not transfer well to other datasets. Additionally, the imbalance between the two classes of data (i.e., malicious and benign) represented in typical network data is not ideal for these machine learning models [25]. More recently, researchers have looked to apply unsupervised learning methods to meet their objectives. Two studies have considered iForest [26-27], a one-class classifier that attempts to overcome the training dataset issues by isolating anomalies inherent in the data, thus removing the need to explicitly inject malicious data into the dataset. However, this approach still requires a cleaned dataset (i.e., only “good” DNS traffic), and requires training a model. In contrast to this, the benefit to a statistical method is (1) it can be applied directly to a dataset as-is, without the need to train a model and determine a sufficient ratio of positive and negative data samples required for training; and (2) the approach is more robust to changes over time, unlike trained models that require retraining at some point to account for different trends in an evolving data stream. One recent study looked at implementing multidimensional data decomposition to detect DNS tunneling and exfiltration. Such a method does not require any prior knowledge of the labels within the dataset, and

showed promise resulting in detection of one case of exfiltration. However, the authors mentioned such an algorithm would require a network security analyst to undergo significant training to gain the expertise needed to interpret results of the model [28].

Our work looks to build on past research by incorporating many of the features investigated to characterize domain-related traffic over the course of a day [23-24] into our analysis. This approach is a simple model that aims to combine both unsupervised methods (i.e., clustering) and statistical analysis (i.e., outlier detection) to detect outliers within the data, without needing prior knowledge of which domains may be benign or malicious within the dataset. Domains flagged as exhibiting exfiltration behavior could be reviewed by a network analyst to determine if the case represents malicious exfiltration. If deemed benign, the domain could be whitelisted to prevent it from showing up as an outlier on a future day. We believe that developing a detection method capable of determining exfiltration based on domain features, without the requirement for a labelled dataset, yields a simple yet effective solution that translates easily to other networks, and provides a novel contribution to the preceding body of research in this field of study.

3 DATASET

In this experiment we used anonymized network data sampled from a real research network at the author's organization. The Lincoln Research Network Operations Center (LRNOC) is a datacenter at MIT Lincoln Laboratory intended to provide access to operational, enterprise network data for research purposes [29]. A vast amount of data consisting of more than 60 unique sources is fed into LRNOC, including DNS logs, raw network packets, alert sensors, and host logs. The data collected provides researchers with a rich, realistic source to test network security algorithms, conduct deep-dive analyses to support investigations of potential malicious behavior, and support the research and development of tools.

Our DNS dataset was collected from LRNOC over a two-week period in March 2017. Over this two week period, there were more than 111,000 unique domains. Because of the sensitivity of this data, the actual domain names will be presented in this analysis using aliases (e.g., "malware.com" as "Domain_1234").

The DNS logs were parsed as follows. For every DNS query in the log, we extracted the query string (e.g. maps.google.com), the source IP, destination IP, and the query record type (e.g. AAAA). Then, for each extracted query string, we separated it into subdomain, domain, and top level domain. (e.g. maps.google.com would have the subdomain "maps", the second level domain "google", and the top level of "com"). The extracted data was then binned according to each set of second level and top level domain pairs (e.g. google.com includes subdomains such as "maps", "docs", and "keep"). Features were extracted from this processed dataset.

Synthetic DNS exfiltration data was also generated, not as input to train a model, but as a way to compare against suspected exfiltration data detected in the analysis. The data was created as follows. Random credit card information was generated using the Python faker package [30]. This data included fake names, expirations, and card numbers. The generated data was then encoded using a base-64 encoding, mimicking the encoded traffic identified in the FrameworkPOS malware attack on Home Depot. The encoded data was added to the subdomain of DNS queries for FrameworkPOS.com (e.g. <encoded credit card information>.FrameworkPOS.com). The record type was assumed to be of type A, since that is the type used in the FrameworkPOS attack. This data was then parsed as normal, and the given features were generated from it.

The following list gives a description of each of the features used, chosen using a combination of the most common features from prior works [23-24]:

1) *Alpha Ratio:* This is the ratio of alphabet characters (A-Z, a-z) for all subdomains of a given domain. Website subdomains tend to be composed of mostly alphabet characters, so a large number of non-alphabet characters could indicate exfiltration.
2) *Average Subdomain Length:* The average length of a subdomain of a given domain. When exfiltrating data, hackers generally want to move data as quickly as possible across the network, so long subdomains can indicate data leaving the network.
3) *Character Entropy:* A measure of the randomness of all the subdomains of a given domain. Since the exfiltrated data is normally encoded, there tends to be a more random assortment of characters in suspicious subdomains.
4) *Digit Ratio:* Similar to Alpha Ratio, this is the ratio of digits (0-9) for all subdomains of a given domain. As previously stated, website subdomains tend to be alphabet heavy, so an influx of numbers in subdomains could indicate exfiltration.
5) *Dot Ratio:* The ratio of '.' characters for all subdomains of a given domain. DNS queries are restricted to 64 bytes per label, which restricts the amount of data an attacker can get out of each query. By adding additional subdomains (separated by '.' characters), attackers can exfiltrate more data in a single query.
6) *Lowercase Ratio:* The ratio of lowercase alphabet characters to other characters. Website subdomains tend to be mostly lowercase. A large ratio of uppercase characters could indicate encoding within the subdomain, since some encoding schemes use both upper and lowercase characters. This might be an indication of exfiltration.
7) *Query Record Type:* The percentage of queries for the given domain with the A or AAAA record type. A and AAAA records are the most common record type, but are restrictive about how big the query can be. Other, less common record types may be used by an attacker to extract data more quickly from a target. A low percentage of A or AAAA record types for a domain might indicate exfiltration.
8) *Unique Query Ratio:* Ratio of unique subdomains to all subdomains for a given domain. Sending encoded data across DNS would generate a lot of unique subdomains, and could indicate exfiltration.
9) *Unique Query Volume:* Total number of unique subdomains for a given domain. Some domains only have one or two total queries in a set of data, which can make the unique query ratio quite high for those domains, even if they are not being used to exfiltrate data. By also using the total volume of requests, we can augment our algorithm to down-weight low query domains.

10) *Uppercase Ratio:* Similar to Lowercase Ratio. A large ratio of uppercase characters could indicate some sort of encoding within the subdomain, since some encoding schemes use both upper and lowercase characters.

Table 1 summarizes the features used in the analysis.

Table 1: Features generated daily for each domain

Feature	Description
1: alp_ratio	# alphabets in subdomain / length of string
2: avg_len	Average length of subdomain
3: char_ent	Total entropy of characters in all subdomains
4: digit_ratio	# of digits in subdomain / length of subdomain string
5: dot_ratio	# of '.' characters in subdomain / length of subdomain string
6: lower_ratio	# of lowercase characters in subdomain / length of subdomain string
7: rr_type	% of queries with A or AAAA record types
8: uni_ratio	# of unique subdomains / total # of subdomains
9: uni_vol	# of unique subdomains
10: upper_ratio	# of uppercase characters in subdomain / length of subdomain string

4 STATISTICAL APPROACH

The features detailed in Section 3 were extracted for each of the domains per day, across the 14-day period. The distributions of these features are shown in Figure 2. These distributions give insight into the expected values for each of the features. A domain performing exfiltration would have outlier values for one or more of these features. Across some of these features, a large majority of the values were equal. Therefore, in order to visualize any variability in the feature, such points were excluded from the histogram plots in Figure 2 but documented in the text label in the graph (e.g. for feature 5, 77% of the "dot_ratio" values were equal to 0, as indicated in the text on the plot; those values were omitted but the remaining values are shown as the distribution in the figure). These features were calculated daily for each domain that appeared in the data for that day, and aggregated into a feature matrix for further analysis.

Principle Component Analysis (PCA) was then applied to the feature matrix so that the features could be visualized graphically. PCA is a dimensionality reduction technique which looks to represent the data in a smaller coordinate system while still preserving the variability among the data points. PCA works by first finding the axis with the highest variability between the data points, and subsequently finding the orthogonal axis in descending order of variability within the data. For our data, PCA was applied using the MATLAB function, and we found that using only the first two principal components explained 99% of the variability within the dataset. An example of the PCA results is shown in Figure 3, where each point represents the features for a single domain represented

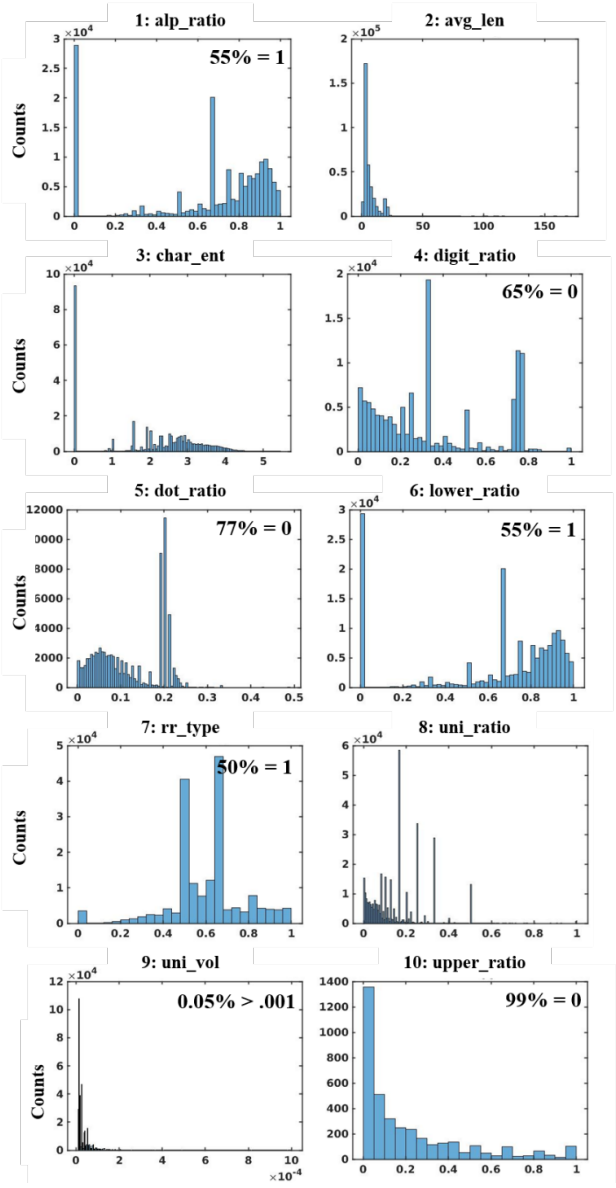

Figure 2: Feature distributions across daily aggregations of DNS query data, for the 14-day period

in the two dimensional principle component (PC) space. An additional observation is the tendency of the data to form denser clusters along near vertical lines. Data points within a group of aligned points tend to share similar values for character entropy and average length of subdomains; therefore, these two features are the largest contributors to this stratification behavior.

While the translation of the data into the PC space provides a way to visualize the data and detect outliers, the relation of the principle components to the original feature set is not necessarily intuitive. Figure 4 strives to address this issue by providing the correlation between each of the features and the two principle components. Here, the blue bars in the top half of the plot correspond to the absolute

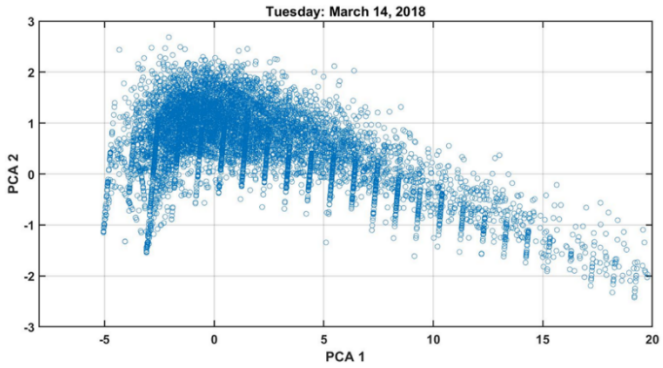

Figure 3: PCA applied to an example day in the dataset

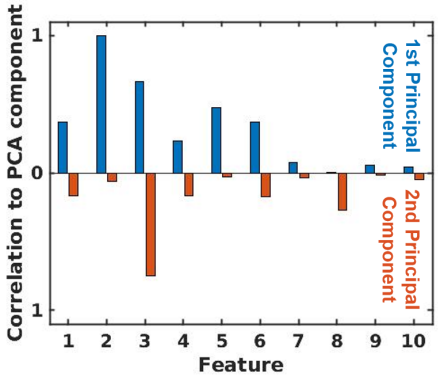

Figure 4: Correlation of each feature to the first and second principal components

correlations between the features and principle component 1. The orange bars in the bottom half of the plot correspond to absolute correlations between the features and principle component 2. It can be seen that a majority of the variance in the second principle component is from the third feature (char_ent). This same feature also contributes significant variance to the first principle component, although the second feature (avg_len) is most correlated with the first principle component. The further a domain is from the centroid of the cloud point, the more likely the subdomain consists of a longer than average string with higher entropy. However, while these two features exhibit the highest correlation to the two principle components, the magnitudes of the correlations for other features seen in Figure 4 indicate non-negligible contributions from most of the remaining features. The features with the least impact appear to be feature 7 (rr_type), feature 9 (uni_vol), and feature 10 (upper_ratio).

The final step before investigating the behavior of the data within the PC space across multiple days is to consider the consequence of continually re-computing the principle components for a new set of DNS data each day. The analysis which is presented in the following sections assumes that the principle components from day to day remain relatively consistent. In order to test the validity of our assumption, we performed a comparison between the PC space

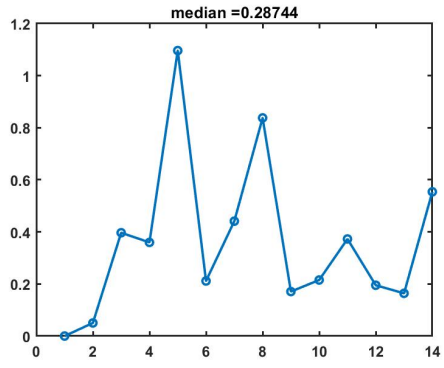

Figure 5: Critical angle for the PC space from day to day over the two week period

between each consecutive set of days during the two week period. Krzanowski [31] outlined a procedure to determine if two sets of principle components are similar or not. This is accomplished by comparing the subspaces defined by the columns of the matrices containing the coefficients of the first k principle components (i.e., in our case, k=2 since we are only considering the first two principal components). A useful and detailed application of this approach is given by Jolliffe [31]. Let L_1 be the matrix where the columns are defined by the first k set of PC coefficients for the first day's data. Let L_2 be the matrix where the columns are defined by the first k set of PC coefficients for the second day's data. Then the minimum angle between the subspaces defined by the columns of L_1 and L_2 is found by:

$$\delta = cos^{-1}\left(\sqrt{\lambda_1}\right) \tag{1}$$

Where λ_1 corresponds to the largest eigenvalue of:

$$\boldsymbol{L_1}'\boldsymbol{L_2}\boldsymbol{L_2}'\boldsymbol{L_1} \tag{2}$$

In a later publication, Krzanowski [33] investigated what constitutes a "small enough" angle such that the two sets of PC components are considered similar. Here multiple simulations were run using different population group sizes, number of variables (p), and number of principal components (k). Our plot in Figure 5 shows angles that are well within the size considered "small", for p=8 and k=2 (p = 10 in our case, so we compare to p=8 which is the closest equivalent to our scenario that is available in the tables from Krzanowki's study [33]). Therefore we can conclude that the PCA axes are similar, and so our original assumption is correct.

4.1 PCA Outliers

The first step towards detecting data indicative of exfiltration is to look at the data within the PC space. Figure 6 shows the DNS query data for the first Tuesday in the two week span of data, plotted in the PC coordinate system. The Mahalanobis distance was used as a means to determine outliers within the two-dimensional space. This metric measures the distances between points in multivariant space, and is commonly applied in statistics to determine outliers. In order to capture the extreme outliers in this analysis, we used a cutoff value of 100, which is greater than the 99.9th percentile of

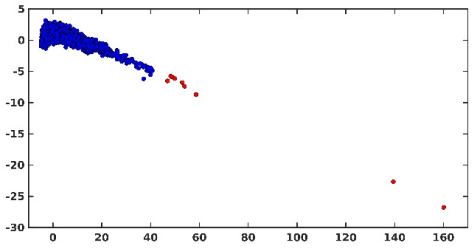

Figure 6: PCA Outliers for an example day (Tuesday of the first week of data)

the Mahalanobis distances across the data represented in Figure 6. The blue points in the plot represent the data within the normal region, and the red points are outliers.

The outlier domains are listed in Table 2 using aliases due to the sensitivity of the network data. The feature values for these points were compared to the feature distributions over the two week period, to get a better understanding of why these domains were flagged as outliers. We observed that for these points, the average length of the subdomain, and likewise the entropy, were all above the 95th percentile. This makes sense, as these were the features that had the highest correlation with the PC space. For the remaining features, the uni_vol was high for the first three domains; the upper ratio was high and the rr_type was low for the 6th domain; and the rr_type was low and the uni_val was high for the 8th domain.

These domains were evaluated to determine if any represented exfiltration. Note we assume (but not guarantee) that the network data is most likely clean of malicious DNS exfiltration occurrences, and therefore the lack of true positives is not indicative that the approach is flawed. The two most prominent outliers are domains for general services lookups, and were similarly found to be false positives in prior studies (e.g. in the work by Nadler et al. [26]). Such services use the DNS protocol as a means to transfer data, but for valid and legitimate purposes. In addition to this specific example day, other days in the two-week span contained outliers that were security services lookups, using the DNS protocol to exfiltrate legitimate data. The remaining domains listed in Table 2 were also found to be legitimate domains. For example, two of the domains are registered to Google and we speculate that they were created as part of a study conducted on security measures for DNS protocol (DNSSEC). Two of the other domains are registered to a legitimate company called Markmonitor, which manages and monitors domain name registration. We did not find any attempts to maliciously extract data during the two week period. However the purpose of the tool is simply to detect exfiltration, so that a network analyst can then determine if the extracted domain is malicious or benign. Therefore, since domains performing legitimate DNS exfiltration were detected using this method, the PCA outlier detection is considered a successful means to extract potentially malicious domains.

In Figure 7, known DNS exfiltration is injected into the dataset and compared to the previously detected outliers. The injected data does not serve to train a model, but is used to compare to the outliers already detected. The exfiltration data was synthetically generated to replicate the encoded data found in the attack on Home Depot, which was accomplished through the malware Framework-POS. The injected data (yellow circles in Figure 7) aligns well with the suspected exfiltration domains (red circles in Figure 7), both highlighted as outliers in the PC space. This provides additional credibility that outlier detection within the PC space is correctly capturing exfiltration domains within the DNS query data.

4.2 Clustering & Arc Diagram

In the previous section, immediate outliers were found simply based on their location within the PC space compared to the majority of the data points. A more complicated detection problem is one where a domain may exhibit more normal behavior on most days, but then demonstrates activity which may be indicative of exfiltration on another day. To address this issue, this section looks to determine a

Table 2: Domains flagged as potentially performing exfiltration using PCA outlier detection

Domain Name	Significant Features	Example subdomain
Domain_5599	avg_len, char_ent, and uni_vol were **high**	p5-zefyqsmpwyesw-h7ky3fy7uaovxnpa-273138-i1-v6exp3-ds.metric
Domain_5598	avg_len, char_ent, and uni_vol were **high**	p5-zefyqsmpwyesw-h7ky3fy7uaovxnpa-273138-i2-v6exp3-ds.metric
Domain_9778	avg_len, char_ent, and uni_vol were **high**	p4-hb7sgm7dysdeo-jwhwnkh3sajsc2qw-228863-i1-bogus-dnssec-vd
Domain_9922	avg_len and char_ent were **high**	p5-25wy5dszgqs5q-yxznimq2jagfegg6-228335-i2-bogus-dnssec-bd
Domain_46787	avg_len and char_ent were **high**	d20b1f1a666e6c4d2f7d9ab5cfae096466d93759.cloudapp-enterprise
Domain_51620	avg_len, char_ent, and upper_ratio were **high**; rr_type was **low**	b45d50cd3293.358778121.AT3776PZPBOVS5DULS5WDNEQTC6OTFF7V54 XZD6EJUBOQAG7LVFQ.6b656015-f0f0-cfbc-4712-01b8b8c815c0.4c4c2d506572736f6e616c2d557365.v1
Domain_51622	avg_len and char_ent were **high**	358778062.6b656015-f0f0-cfbc-4712-01b8b8c815c0.b45d50cd3293.4c4c2d506572736f6e616c2d557365.b0
Domain_45234	avg_len, char_ent, and uni_val were **high**; rr_type was **low**	c3aiaakis23hpnb3q4c5bimsbffntdw55kj4irtb.ktlwnu5pflxhv3smie3eqqkx5s35 qfqlrx32v3tpqi3pwaw5cvm5vs5sxx2ihqe.5cvwvwmvc6xnlnl3dkabpqi5zp6t3 ahlemwo22m6qd2zxfnlgpx7i4vl6wrx2oa
Domain_52336	avg_len and char_ent were **high**	d2a9c7fbc3cc4b34-76633a61cf4e4028bcadb8b43290649f.orf

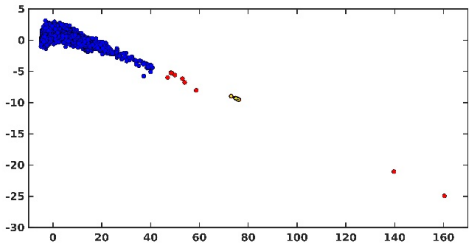

Figure 7: PCA Outliers for an example day (Tuesday of the first week of data). Generated DNS exfiltration data is shown as the yellow stars

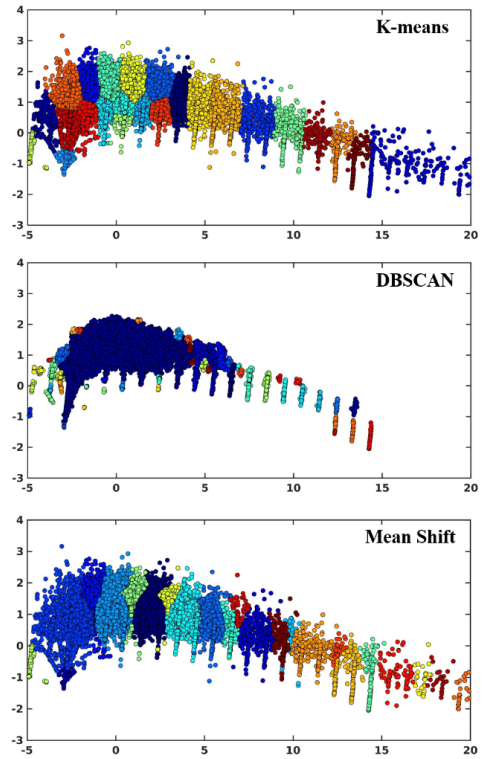

Figure 8: Results from clustering using K-means (top), DBSCAN (middle), and Mean Shift (bottom) algorithms

baseline for a domain's expected location, compared to where the domain is actually found within the PC space. To detect such an occurrence, this section considers clustering each day's worth of data and evaluates which cluster a domain typically belongs to.

As mentioned previously, the data within the PC space shows a point cloud where points tend to be more densely populated along vertical lines. Different methods were tested to cluster the data, but these did not adequately capture the vertically stratified behavior of the data. The results from clustering attempts using K-means [34], DBSCAN [35], and Mean Shift [36] algorithms are shown in Figure 8.

The result of the K-means algorithm is shown in the top panel of Figure 8. For the K-means algorithm, the number of clusters are chosen prior to implementation of the clustering method. For this analysis, k was chosen to be 24, to match the number of vertical groups seen in the data. Initially, centroids are assigned for each of the k clusters. The remaining points are grouped to the clusters with the closest centroid based on Euclidian distance. The centroids are then recalculated for each of the clusters. This process is repeated until the centroids converge. Since this method looks to minimize the distance between the mean of each cluster and its members, points within a point cloud will tend to form more circularly shaped clusters, and will not naturally conform to the stratified behavior exhibited by the DNS query data in the PC space.

The result of the DBSCAN algorithm is shown in the middle panel of Figure 8. This method works by identifying two types of points in the data: "core points" and "directly reachable" points. Core points are points that contain a minimum number of points within some radius ϵ. The minimum number of points and ϵ are inputs to the algorithm. For this analysis, ϵ was set to .1 and minimum number of points to 10. A directly reachable point is a point within ϵ of a core point; note that some or all of these points may be simultaneously considered core points as well. Points that do not fall into either of these categories are considered outliers. A cluster is then defined by neighboring core points, and directly reachable points. A cluster must have at least one core point, but does not have a maximum number of core points. Therefore, in a point cloud where points are dense enough, this algorithm will consider a large group of points to be a single cluster, rather than identifying smaller vertical bands as individual clusters within that group.

Finally, the result of the Meanshift algorithm is shown in the bottom panel of Figure 8. In this method, a window surrounding each point in the data is defined using Euclidean distance. For each window, the weighted mean of all the points within the window is calculated using a Kernel, such that the contribution of points from the main point within that window decays exponentially with distance. This calculated mean is used as the new center of the window for step 2. Points in the vicinity, following the step just described, may converge to the same point for step 2; therefore after one iteration there are now less points than the total number of datapoints. The initial step is repeated with only the newly calculated points, and this process continues until the calculated points between steps do not change. Data points that have converged to the same final point are deemed to be a cluster. While this method is an improvement over the results observed from both K-means and DBSCAN in capturing more vertically aligned clusters, the results shown in the figure still highlight the need for an algorithm that more finely captures the individual vertical bans of density.

In order to group points based on the vertically dense regions, we took a different approach. We binned the values of each data point in the first principle component, and the peaks of the binned data were used to find the location of the denser lines within the point cloud. This is shown in the top and middle plots of Figure 9. The edges of each group were simply defined as the midpoint between peaks. The resulting clusters can be seen for an example day's data in the bottom plot of Figure 9. This technique was then applied to each individual day within the two week period.

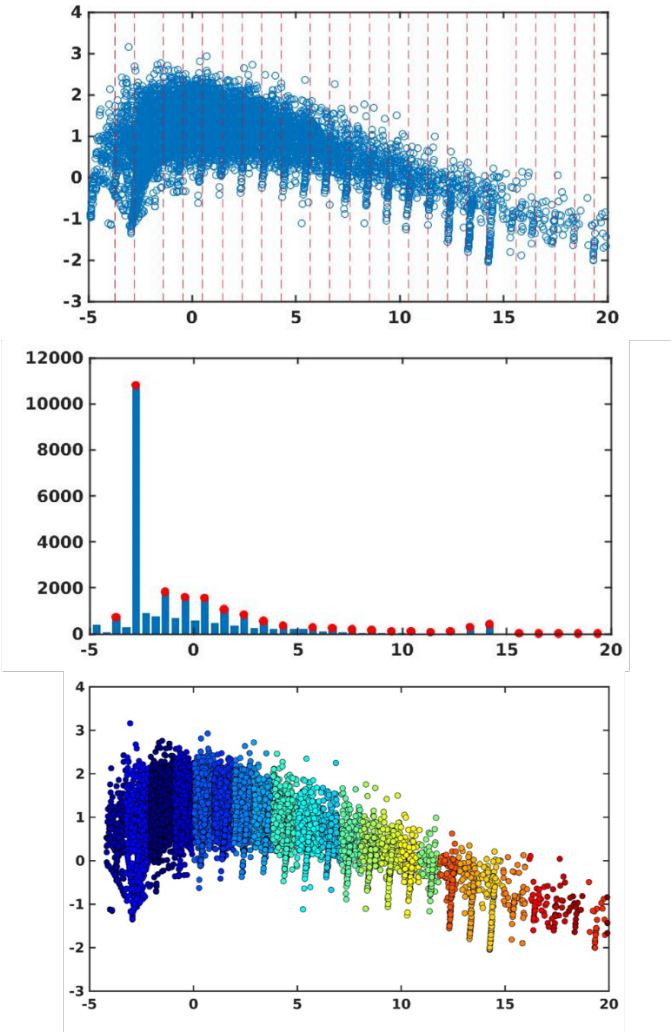

Figure 9: Identifying the densely populated lines within the data and the clustering results

With the clustering complete, our goal was to utilize the identified clusters between consecutive days to extract domains that exist within the confines of the "normal" data, but which might actually be an outlier based on the cluster that the domain would typically be grouped with compared to the cluster in which it is found. An example for this reasoning would be the case where an attacker waits until the weekend when network security analysts are less likely to watch the data stream to then exfiltrate the largest amount of data. To visualize such an outlier on a given day, we employed an arc diagram. An arc diagram is a specific type of network plot, where the nodes of the network are aligned along a single axis [37]. In the case of the DNS data, a node represents a cluster on one of the two days, where the size of the node is proportional to the number of domains within that cluster. For annotation purposes, let the node for the i^{th} cluster on day 1 be define as n_i^1; and, let the node for the j^{th} cluster on day 2 be defined as n_j^2. An edge between

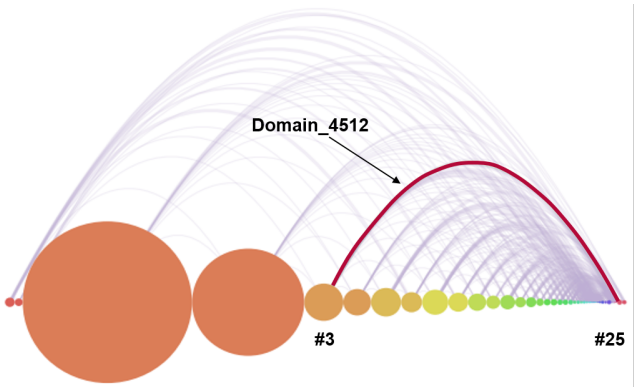

Figure 10: Arc diagram for an example 2-day period in the dataset

a node A and a node B means that at least one domain that existed in cluster A on day one exists in cluster B on day two. Because our goal is to pinpoint domains that switched cluster affiliation, the edges are weighted by the distance between the clusters, divided by the number of domains which also switched between the same clusters:

$$e = \frac{abs\left(n_j^2 - n_i^1\right)}{\#\ of\ domains\ in\ n_j^2\ from\ n_i^1} \tag{3}$$

The visualization was implemented in python using the open source library NetworkX [38]. Figure 10 shows the results for the first Saturday and Sunday in the dataset. The highlighted edge is the one with the highest weight for the two day period.

This domain existed in cluster 3 on Saturday, and jumped to cluster 25 on Sunday. This domain appeared only a handful of times on the first day, and the subdomain was either 'm' (indicative of access from a mobile device) or 'www'. On the following day, the domain appeared over 150 times, with multiple subdomains consisting of long strings with an ambiguous collection of characters which did not represent English words. The avg_len, char_ent, uni_ratio, and uni_vol values were all high for this domain on the second day, but were within normal ranges on the preceding day. Table 3 summarizes the significant features for this domain, and provides an example of one of the suspect subdomains. The domain is owned by a well-known legitimate gaming site; we suspect that the suspicious data seen within the subdomains is benign information, such as the transfer of user data. This behavior is an example of legitimate use of the DNS protocol. A number of applications use the protocol for quasi-legitimate purposes; Spotify is a documented example [39].

As a final sanity check, Figure 11 shows a heatmap for the 2-week period, where the x-axis corresponds to cluster numbers on a given day and the y-axis the cluster numbers for the following day. The intensity of each of the points indicates the probability that a domain within a given cluster on the first day is found in the same cluster the following day. The graph shows a high intensity along the diagonal, which represents domains that stay within the same cluster on consecutive days. This means that traffic typically remains within a given cluster across different days, and validates the assumption that the DNS traffic remains mostly consistent

Table 3: Domain flagged as potentially performing exfiltration using the clustering method

Domain	Significant Features	Example Subdomain
Domain_4512	avg_len, char_ent, uni_ratio, and uni_vol were **high**	c-7npsfqifvt34x24gpoutx2ehpphmfbqjtx2edpn.g00

Figure 11: Average change in cluster assignment for domains

within the PC space. This assumption is critical to the ability to detect outliers with the method presented above.

5 PIPELINE FOR ANALYST

The different analytical techniques discussed in Section 4 can be aggregated into a single tool for use by a network analyst. Such a capability would function by implementing the three different outlier detection techniques described in the preceding section, on the previous day's DNS data. Suspicious DNS domains flagged by any of the three techniques would be aggregated and passed initially through a whitelist filter to remove known benign domains (i.e., spotify.com), before they are written to a text file or displayed to an analyst for further review. Upon investigation of each domain, the analyst could either (1) whitelist the domain if it is determined to be benign, preventing it from passing through the filter at a later time or (2) block the domain if it is determined to be malicious exfiltration. The source IP address(es) of the query could then be used to determine which computer on the network has been infected. The supporting visualizations for each analytic, along with the raw DNS traffic for every suspicious domain, could be included in an additional log file to expedite the investigation process for the analyst. Figure 12 summarizes the pipeline for such an application.

6 CONCLUSIONS AND FUTURE WORK

This analysis investigated the development of statistical methods and unsupervised learning methods to detect DNS queries within network data exhibiting low-throughput exfiltration. Such an approach does not require a labelled or curated dataset (as is the case

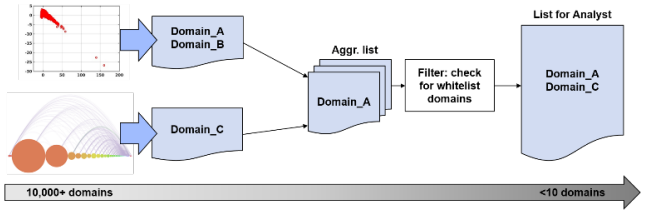

Figure 12: Pipeline for implementing the tool

with supervised or one-class machine learning techniques), can be applied directly without first needing to train a model, and is robust to changes in the trend of the network traffic over time as the analytics are recomputed on a daily basis. The final output is a tool which is simplistic enough for a network analyst to decipher without any special background, yet still effective.

Future work could include the development of a more robust clustering method that is not dependent on binning the data in one dimension within the PC space – such an approach can be sensitive to the number of bins chosen during the processing. Additionally, incorporating features on the domain name, instead of only the subdomain, could provide further insight as to whether a domain is performing malicious or benign exfiltration. Consideration of other features (e.g. the ratio between the longest meaningful word and the subdomain length [26]), along with a more in-depth evaluation of current features to remove any that are redundant, will help capture variability among data points efficiently. Finally, extending the analysis to data encompassing a much longer period of time, and evaluating results of using both the proposed method and an alternative supervised approach on this dataset from, would provide insight into the level of robustness achieved from our model as compared to the supervised method.

ACKNOWLEDGMENTS

We would like to thank Isidore Venetos and Vidyut Patel of the FAA NextGen Aviation Research Division (ANG-E2) for their valuable feedback on this program.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Federal Aviation Administration under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Federal Aviation Administration.

by the U.S. Government may violate any copyrights that exist in this work.

REFERENCES

[1] Palo Alto Networks. What Is DNS Tunneling? Retrieved from https://www.paloaltonetworks.com/cyberpedia/what-is-dns-tunneling.

[2] Asaf Nadler and Avi Aminov. The Akamai Blog. Introduction to DNS Data Exfiltration. Retrieved from https://blogs.akamai.com/2017/09/introduction-to-dns-data-exfiltration.html

[3] Plixer. 2015. Security Vendors Teaching Bad Actors How to Get Past Firewalls. Retrieved from https://www.plixer.com/blog/security-vendors-teaching-bad-actors-how-to-get-past-firewalls/

[4] G DATA. 2014. New FrameworkPOS variant exfiltrates data via DNS requests. Retrieved from https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests

[5] Krebs on Security. 2015. Deconstructing the 2014 Sally Beauty Breach. Retrieved from https://krebsonsecurity.com/2015/05/deconstructing-the-2014-sally-beauty-breach/

[6] Lance Whitney. TechRepublic. 2019. How organizations can better defend against DNS attacks. Retrieved from https://www.techrepublic.com/article/how-organizations-can-better-defend-against-dns-attacks/

[7] Sourceforge. 2003. NSTX. Retrieved from http://nstx.sourceforge.net/

[8] Alessio Merlo, Gianluca Papaleo, Stefano Veneziano, and Maurizio Aiello. 2011. A Comparative Performance Evaluation of DNS Tunneling Tools. In *International Conference on Computation Intelligence in Security for Information Systems*. DOI:https://doi.org/10.1007/978-3-642-21323-6_11

[9] Tadeusz Pietraszek. 2004. DNSCat. Retrieved from http://tadek.pietraszek.org/projects/DNScat/

[10] Iodine. 2010. Retrieved from http://code.kryo.se/iodine/

[11] Lucas Nussbaum, Pierre Neyron, and Olivier Richard. 2009. On Robust Covert Channels Inside DNS. In Gritzalis D., Lopez J. (eds) Emerging Challenges for Security, Privacy and Trust. SEC 2009. IFIP Advances in Information and Communication Technology, Vol. 297. Springer, Berlin, Heidelberg

[12] Olivier Dembour. DNS2TCP. Retrieved from https://github.com/alex-sector/dns2tcp

[13] Ozyman, Retrieved from http://www.cship.info/mirror/dnstunnel/ozymandns_src0.1.tgz

[14] Greg Farnham. 2013. Detecting DNS Tunneling. In *SANS Institute*. Retrieved from https://www.sans.org/reading-room/whitepapers/dns/paper/34152.

[15] Christian J. Dietrich, Christian Rossow, Felix C. Freiling, Herbert Bos, Maarten van Steen, Norbert Pohlmann. 2011. On botnets that use DNS for Command and Control. In *Proceedings IEEE Computer Network Defense*, Gothenburg, Sweden, pp.9-16

[16] Maurizio Aiello, Maurizio Mongelli and Gianluca Papaleo. 2015. DNS tunneling detection through statistical fingerprints of protocol messages and machine learning. In *International Journal of Communication Systems*, Vol 28(14), pp. 1987-2002.

[17] Maurizio Aiello, Maurizio Mongelli, Enrico Cambiaso and Gianluca Papaleo. 2016. Profiling DNS tunneling attacks with PCA and mutual information. In *Logic Journal of the IGPL*, Vol 24(6), pp. 957-970.

[18] Kenton Born and David Gustafson. 2010. Detecting DNS Tunnels using Character Frequency Analysis. In *Proceedings of the 9th Annual Security Conference*, Las Vegas, USA.

[19] Anna L. Buczak, Paul A. Hanke, George J. Cancro, Michael K. Toma, Lanier A. Watkins, and Jeffrey S. Chavis. 2016. Detection of Tunnels in PCAP Data by Random Forests. *In CISRC '16: Proceedings of the 11th Annual Cyber and Information Security Research Conference.*

[20] Maurizio Aiello, Maurizio Mongelli and Gianluca Papaleo. 2014. Supervised Learning Approaches with Majority Voting for DNS Tunneling Detection. In *de la Puerta J.* et al. *(eds) International Joint Conference SOCO'14-CISIS'14-ICEUTE'14. Advances in Intelligent Systems and Computing*, Vol 299. Springer, Cham.

[21] Jiacheng Zhang, Li Yang, Shui Yu and Jianfeng Ma. 2019. A DNS Tunneling Detection Method Based on Deep Learning Models to Prevent Data Exfiltration. *In Liu J., Huang X. (eds) Network and System Security. NSS 2019. Lecture Notes in Computer Science*, Vol 11928. Springer, Cham

[22] Yakov Bubnov. 2018. DNS Tunneling Detection Using Feedforward Neural Network. *In European Journal of Engineering Research and Science*, Vol 3(11).

[23] Anirban Das, Min-Yi Shen, Madhu Shashanka, and Jisheng Wang. 2017. Detection of Exfiltration and Tunneling over DNS. In *16th IEEE International Conference on Machine Learning and Applications.*

[24] Jawad Ahmed, Hassan Habibi Gherakheili, Qasim Raza, Craig Russell, and Vijay Sivaraman. 2019. Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks. In *Proceedings of IFIP/IEEE IM.*

[25] Bin Yu, Les Smith, Mark Threefoot and Femi Olumofin. 2016. Behavior Analysis based DNS Tunneling Detections and Classification with Big Data Technologies. *In Proceedings of the International Conference on Internet of Things and Big Data*, pp. 284-290.

[26] Asaf Nadler, Avi Aminov, Asaf Shabtai. 2019. Detection of malicious and low throughput data exfiltration over the DNS protocol. In *Computers & Security*, Vol. 80, pp. 36-53.

[27] Jawad Ahmed, Hassan H. Gharakheili, Qasim Raza, Craig Russel, and Vijay Sivaraman. 2019. Monitoring Enterprise DNS Queries for Detecting Data Exfiltration from Internal Hosts. In *IEEE Transactions on Network and Service Management.*

[28] David Bruns-Smith, Muthu M. Baskaran, James Ezick, Tom Henretty. Richard Lethin. 2016. Cyber Security through Multidimensional Data Decompositions. In IEEE 2016 Cybersecurity Symposium (CYBERSEC).

[29] Lincoln Research Network Operations Center. 2020. Retrieved from https://www.ll.mit.edu/about/facilities/lincoln-research-network-operations-center

[30] PiPy. 2016. Faker. Retrieved from https://pypi.org/project/Faker/0.7.3/

[31] W. J. Krzanowski. 1979. Between-Groups Comparison of Principal Component. In *Journal of the America Statistical Association*, Vol. 74, No. 367, pp. 730-707

[31] I.T. Jolliffe. 1986. Principal Components in Regression Analysis. In: *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, NY.

[33] W. J. Krzanowski. 1982. Between-group comparison of principal components – some sampling results. In *Journal of Statistical Computation and Simulation*, 15:2-3, pp. 141-154.

[34] J. A. Hartigan and M.A. Wong. 1979. A K-means Clustering Algorithm. In *Journal of the Royal Statistical Society*. Series C (Applied Statistics), Vol. 28 pp. 100-108.

[35] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd Acm International Conference on Knowledge Discovery and Data Mining* (KDD). 226-231.

[36] Keinosuke Fukunaga and Larry D. Hostetler. 1975. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. In *IEEE Transactions on Information Theory*, Vol. 21(1), pp. 32-40.

[37] Yan Holtz. Retrieved from https://www.data-to-viz.com/graph/arc.html

[38] PyPi. 2020. Nxviz. Retrieved from https://pypi.org/project/nxviz/

[39] Dennis Tatang, Florian Qunkert, Nico Dolecki, and Thorsten Holz. 2019. A Study of Newly Observed Hostnames and DNS Tunneling in the Wild. Retrieved from https://arxiv.org/abs/1902.08454