DNS Tunneling Detection with Supervised Learning

Richard Preston
The MITRE Corporation, Bedford, MA
and Tufts University, Medford, MA
rhpreston@mitre.org

***Abstract*—This paper presents an advanced analytic capable of detecting general DNS tunneling behavior with high precision and recall. It explores the application of supervised machine learning to a recently introduced technique for analyzing DNS traffic: classifying primary domains instead of queries. This approach is enabled by a partially synthetic dataset generated with a structurally configurable DNS tunneling tool.**

Keywords—DNS, DNS tunneling, threat detection, data analytics, machine learning, traffic generation

I. Introduction

The Domain Name System (DNS) is commonly referred to as the phonebook of the Internet. When you type "www.ieee.org" into your browser, a query is sent to a DNS resolver to determine the location, or IP address, of the domain. This happens for most online activity, and therefore, DNS traffic is prevalent on all networks that connect to the Internet. Unfortunately, the DNS protocol is fraught with security vulnerabilities, and efforts to fix the 1980s technology have largely failed [1]. According to a recent survey, 33% of organizations suffered data theft via DNS in 2018, costing an average of $715,000 per incident [2]. The combination of a colossal legacy system and sophisticated evolving threats means more advanced approaches are needed to protect enterprises from attackers.

One of the most glaring and difficult problems with the DNS is that it can be hijacked to transmit arbitrary data. The DNS does not consist of a single database of all the IP addresses in the network. Instead, it is a distributed, dynamic system allowing new domains to be registered as they come online. This means that it is trivially easy to add a new name server that you control to the system. For example, if you register the domain "evil.com", then any new queries to "<subdomain>.evil.com" will be directed to your server. Then, any computer in the world can send a message to you by making a DNS query with the data encoded in the subdomain field. The server can respond with any resource record, including IP addresses or arbitrary text. Fig. 1 illustrates this threat, which is known as DNS tunneling.

Over the last decade, academic and industry research has produced techniques for identifying and responding to DNS tunneling [3]. However, detection of evasive protocols has proven difficult and is only now being integrated into commercial devices and software, where the design is often obscured, and the solutions can be prohibitively expensive. In June 2018, Nadler et al. introduced a new approach that enables detection of low-throughput data exfiltration over DNS [4]. The work presented here reproduces and extends theirs by applying supervised machine learning to similar statistical features extracted from a partially synthesized dataset representing a broad spectrum of DNS tunneling protocols. This effectively improves the precision and recall of the classifier without overfitting to a particular malware family.

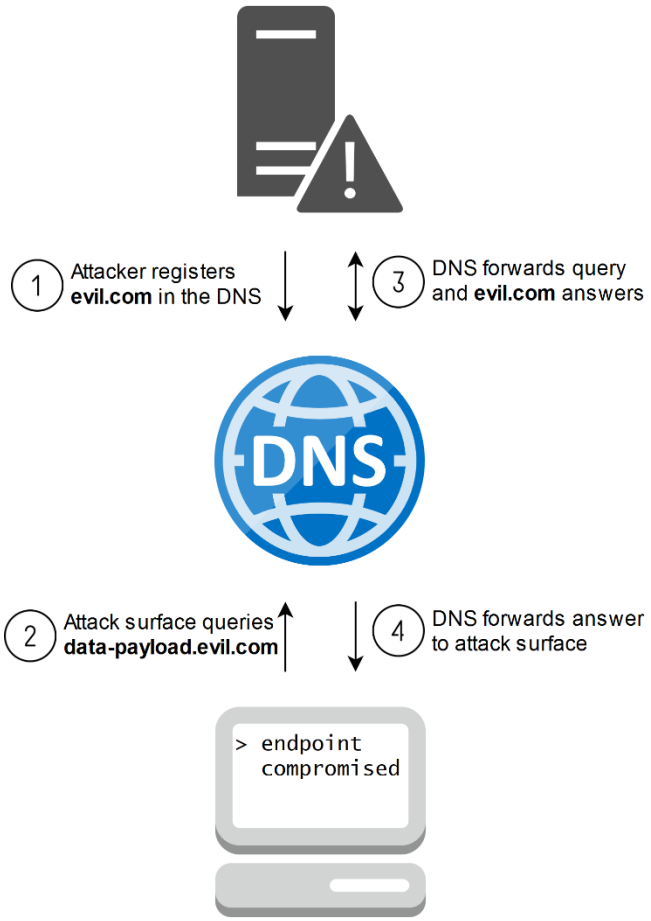

Fig. 1. Simplified example of how DNS tunneling can be used as a communication channel. Note the outgoing data payload is contained in the subdomain field of the DNS query.

II. Approach

A. Background

The challenge in detecting DNS tunneling behavior in general is that each malware family may behave differently, using its own application-layer protocols and performing a distinct set of activities. However, there are three common threads:

1. Malicious actors use DNS tunneling to transmit information.
2. The outgoing information is encoded in the subdomain field of the DNS queries.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

3. Two or more queries that are part of a transmission between a client and server must contain the same primary domain (e.g. “evil.com”).

Obviously, statistical metrics highlighting information content of DNS queries are informative, but the salient ingenuity of Nadler et al. comes from an understanding of point 3. They proposed that the queries be first grouped by primary domain, and then features extracted from the list of subdomains contained in each group. In other words, attempt to classify malicious *domains* instead of malicious *queries*. Framing the problem this way grants more flexibility in the machine learning features that can be extracted, as discussed in the next section.

B. Machine Learning Features

1) Character Entropy: Information (Shannon) entropy as shown in (1) applied to the subdomains taken as a string of characters.

$$H(X) = -\sum_{x_i \in \Omega} P_X(x_i) \cdot log\, P_X(x_i) \quad (1)$$

$P_X(x_i)$ is the normalized frequency of x_i in X, or the number of occurrences of x_i in X divided by the number of elements in X. In this case, X is the concatenation of the subdomain fields for a given list of queries grouped by primary domain, and Ω is the set of characters seen therein.

This feature is intended to measure information content, and therefore domains involved in DNS tunneling are expected to exhibit a higher character entropy.

2) Alphanumeric Content Ratio: The number of lowercase alphabetic or numeric characters (the characters officially sanctioned in the DNS protocol) in *X* divided by the number of characters in *X* (where *X* is as specified in *Character Entropy*).

Normal DNS traffic should only use alphanumeric characters, so this feature may highlight odd behavior such as the use of base64 encoding.

3) Unique Query Volume: The number of unique queries in the group, normalized with (2), where *x* is the unique query volume, and *c* is equal to 20.

$$f = 1 - e^{-\frac{x}{c}} \quad (2)$$

The more queries are sent, the more information can be transmitted. Therefore, a high unique query volume may correlate with DNS tunneling.

4) Unique Query Ratio: The number of unique queries divided by the total number of queries in the group.

For typical use of DNS, identical queries may be made over time to access the same resources repeatedly. With DNS tunneling, the queries are expected to vary in order to transmit new data and to prevent DNS caching.

5) Average Subdomain Length: The average number of characters (bytes) in the subdomains, normalized with (2), where *x* is the average subdomain length, and *c* is equal to 50.

The longer the subdomain is, the more data can be encoded therein.

6) Average Longest Meaningful Word Length Ratio: The longest meaningful word length ratio is the length of the longest English word contained in a string divided by the length of the string. This was computed for each subdomain in the group, and the average was taken as the feature.

Typical normal subdomains are English words, but those used for DNS tunneling are not likely to be meaningful to a human.

7) Average English Content Ratio: The intention of this feature is to estimate the proportion of characters in the subdomains that compose English words, and therefore are unlikely to be used for data transmission. The heuristic in Fig. 2 was used to determine the English Content Ratio (ECR) of each subdomain. The feature was taken as the average ECR over all the subdomains in the group.

8) Average Similarity Ratio: The similarity ratio between two strings is the number of characters they share divided by the number of characters in total. For example, the simality ratio between “abcd” and “cdef” is 0.5. Note that only non-overlapping matching subsequences are considered when counting shared characters. For example, the ratio between “abcd” and “cdab” is still just 0.5.

Rather than compute this for every pair of subdomains in the group, a “sliding window” approach was used. For each subdomain, the average of the similarity ratios between it and the subsequent N subdomains in the list were taken. (In the trials discussed in this paper, N=20.) These values were averaged to obtain the feature.

This is perhaps the most interesting statistic, because it gets to the heart of the difference between normal use of DNS, where subdomains follow some easily-recognizable pattern, and DNS tunneling, where each subdomain represents a data transmission packet and may be totally unrelated to others under the same primary domain.

C. Normal DNS Data

The University of Southern California’s LANDER project annually collects large volumes of Internet traffic as part of their Day in the Life of the Internet effort [5]. A sample of DNS queries requesting A, AAAA, and TXT records was taken from their 2018 dataset to represent normal DNS behavior. To mitigate the concern that the sample may be contaminated by a small portion of DNS tunneling traffic,

```
1   algorithm ECR is
2   input: string S
3   output: number that is the ECR of S
4
5   M := predefined minimum word length
6   L := length of S
7   while True do
8       W := longest meaningful word in S
9       if length of W is less than M break
10      remove W from S
11  return 1 – (length of S) / L
```

Fig. 2. Greedy algorithm implementing ECR approximation. This design emphasizes performance and simplicity over accuracy.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

anomalies were filtered out with an Isolation Forest classifier using the features from the previous section, and the remainder was used for supervised learning. In the end, over 30 million DNS queries were paired down and preprocessed to make up six sets of ~80,000 benign-domain feature vectors each. The data was broken up this way to enable repeated trials when testing the efficacy of the approach.

D. DNS Tunneling Tool

A typical route for recording malicious traffic is to run one or more malware samples in a controlled environment and watch their behavior. This method was explored but abandoned for the simple reason that known malware families and tunneling protocols may not be representative of future attacks. Instead, a tool was developed in Node.js to perform DNS tunneling over a local network and simulate domains exhibiting various behaviors. This exercise provided the following insights into the adversarial perspective and some of the key constraints when working inside the DNS protocol:

1) Length Restrictions: DNS queries cannot be arbitrarily long. A single QNAME is limited to 253 bytes, including label separators (periods). Messages longer than this must be broken into multiple packets. An adversary may also limit the number of bytes used to avoid detection and decrease the risk of dropped packets. DNS also limits the number of bytes in each label to 63. This complicates the process of sending a datagram as a DNS query, because periods must be inserted every 63 characters, which in turn increases the total query length.

2) Limitations to UDP: DNS queries are typically served over UDP, in which IPv4 packets are limited to 512 bytes (on some systems). This is not a problem for the client, since a QNAME is limited to 253 bytes. However, the server must be aware of how many bytes its answer is adding before sending the response, or a CONNREFUSED error may be triggered. In other words, there is a limit to how much data can be encoded in a server reply; multiple queries are necessary when dealing with large server-to-client transmissions. This problem is resolved in the wild by switching to TCP if more than 512 bytes are needed. It can also be mitigated with the use of message compression, which uses the space in a DNS packet more efficiently [6]. (The tool did not support TCP or message compression.)

3) Valid Characters: Restrictions are imposed on what characters can constitute a valid hostname [6]. Specifically, only letters, digits, and hyphens are allowed (with no hyphen at the beginning of a label). However, this is only technically enforceable at the domain registration level, meaning that subdomains (the portion of the FQDN carrying the data in DNS tunneling) might get away with other characters. The risk is that lack of support or deliberate filtering by DNS infrastructure between the client and server may cause packets to be dropped.

4) Caching: Many layers of potential caching exist in the DNS (i.e., places where the result of a query may be saved and retrieved upon subsequent identical queries). Therefore, each query must be unique to ensure it arrives at the server. The tool accomplishes this by inserting a random English word of pre-specified length into the subdomain.

5) Client-only Originiation: Due to the nature of the DNS protocol, all transmissions must conform to a question-and-answer format. In other words, the client must originate all communication with the server. This can be tricky to work around; in some cases, the client must simply poll the server until it receives an indication to stop. Fig. 3 offers a small window into how this works in practice.

The tool was designed for many uses and configurations so that it could synthesize a robust and diverse dataset. The following client behaviors (commands) were supported:

- Ping the server.
- Send a short message to the server.
- Send a short message to the server and receive the same message as a response.
- Send a long message to the server.
- Transmit a file to the server.
- Retrieve a file from the server.

The following parameters were varied in the simulation:

- Command (ping, short message, etc.).
- Resource record type (A, AAAA, TXT).
- Character encoding (base64, base32, hex, utf8).
- Encryption (AES or none).
- Message length.
- Number of commands per domain.
- File transmitted (empty, text, image, etc.).
- Length of anti-caching string.

A simulation was designed to synthesize DNS traffic for domains exhibiting each of the supported behaviors with randomly chosen configurations. A single run produces 4,800 domains, each having a different number of queries. Six runs were conducted to complement the benign-domain feature sets, for a total of about 4 million DNS tunneling queries.

E. Threat Classification

The parsing and feature extraction infrastructure was written in Python so that the application could make use of the excellent scikit-learn library [7]. This design choice enabled the exploration of six machine learning algorithms: Random Forest, Gradient Boosting, AdaBoost, Bagging, Support Vector Classifier (SVC), and Stochastic Gradient Descent (SGD). For each, 80% of the preprocessed and labeled dataset was used for training and validation, and 20% was held out for testing.

```
[server.js]: DNS server [redacted] started on port 53
[client.js]: Querying bfgrailoa.evil.com
[server.js]: Received DNS query - bfgrailoa.evil.com
[server.js]: Decoded payload - p
[server.js]: Ping received
[server.js]: Sending response - OK
[client.js]: Received response - ["0.0.0.0"]
```

Fig. 3. Combined log outputs of DNS tunneling tool resulting from starting the server “evil.com” and executing the ping command from the client.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

The hyperparameters of the estimators were selected with randomized parameter optimization and 3-fold cross validation using the "F1" score (the weighted harmonic mean of precision and recall). The final results were determined by the performance of the classifiers on the hold-out set.

III. Results

A. Feature Value Distribution

Fig. 4 gives a rough picture for each of the features of how the distributions of benign and malicious samples differ. It shows that most are quite good at distinguishing benign and malicious traffic, except for those that try to analyze the content of the query based on an expectation of what normal DNS traffic is "supposed" to look like (Alphanumeric Content Ratio, Average LMW Length Ratio, and Average English Content Ratio). This suggests that our intuitions about the typical use of DNS may not exactly match the reality.

It bears noting that the Unique Query Ratio (UQR) plot contains a spike of benign domains at 0.5. There are several potential reasons for this phenomenon, including redirects, a single-retry policy, or domains with exactly two total queries. Regardless, it represents a potential vulnerability with this feature, as a malicious actor might mask itself by tuning its UQR to 0.5.

B. Classifier Performance

The gathered datasets allowed for six tests. The following hyperparameters were selected based on the validation step for one of the six datasets (the same were used for all tests to be consistent):[1]

- *AdaBoost:* n_estimators=175, learning_rate=1.5
- *Bagging:* n_estimators=75, max_samples=1.0, max_features=6
- *GradientBoosting:* n_estimators=200, max_features=None, max_depth=5, loss=exponential, criterion=friedman_mse
- *RandomForest:* n_estimators=100, max_features=None, max_depth=10, criterion=entropy, class_weight=None
- *SVC:* shrinking=False, kernel=rbf, gamma=scale, class_weight=None, C=1.05
- *SGD:* tol=0.1, penalty=l1, max_iter=200, loss=hinge, class_weight=None

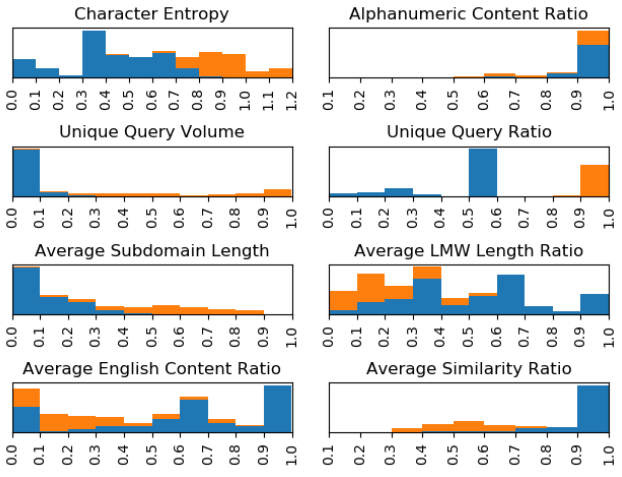

Fig. 4. Distrubution of feature values for benign and malicious samples. The charts are histograms with bins of width 0.1. Blue represents benign samples, and orange represents malicious samples.

Fig. 5 shows the average precision and recall for each classifier over all the trials. In this case, precision is the portion of domains classified as malicious that actually came from the DNS tunneling dataset, and recall is the portion of actually malicious domains that were classified correctly. In other words, precision measures the confidence of the model when it flags a domain as malicious, and recall measures how likely the model is to detect a malicious domain when it is given one.

These results indicate that the application of supervised learning to this problem greatly improves the performance of the analytic. For comparison, the same tests were run for the Isolation Forest classifier, the unsupervised learner used by Nadler et al. It achieved an average precision of 0.8834 and an average recall of 0.7992 with the following hyperparameters:

n_estimators=175, max_samples=0.0625, max_features=4, contamination=0.05

The training time for each trial was also recorded. Fig. 6 shows the average runtime for the training step of each estimator. When selecting a machine learning algorithm to use in an operational analytic, the tradeoff between fidelity and speed must be considered. For example, if training is conducted often and with large datasets, SGD may be a better choice even though it exhibits lower recall.

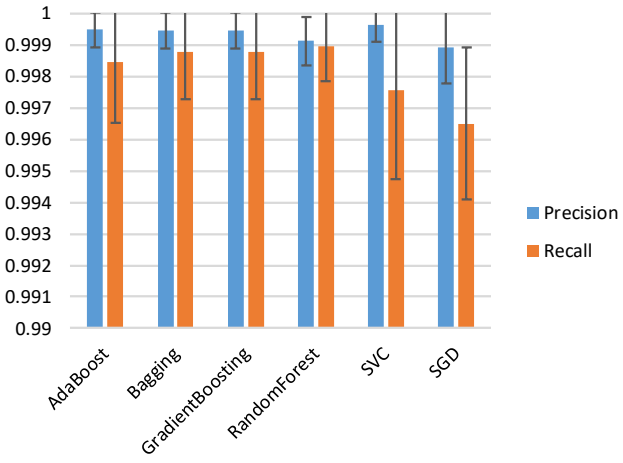

Fig. 5. Average precision and recall over six trials. Error bars represent one standard deviation.

[1] See scikit-learn.org for documentation of each parameter.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

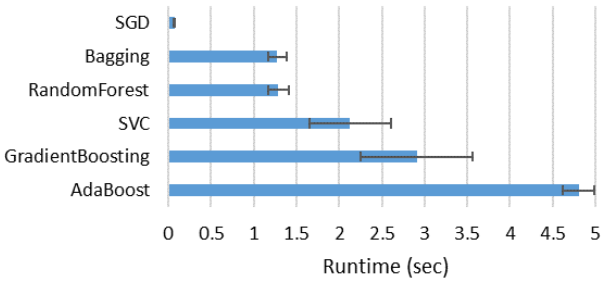

Fig. 6. Average execution time for training step over six trials, with worst case removed. Error bars represent one standard deviation.

C. *Efficacy Confirmation*

To verify the benign set was representative of queries in an operational environment, another trial was conducted with confidential enterprise DNS traffic instead of the USC sample. Though there were some small differences in the feature value distribution (i.e., Fig. 4 looked slightly different), the classifiers exhibited only a marginal performance degradation.

Perhaps a greater concern is that the DNS tunneling tool is not diverse enough to capture the behavior of real malware. To validate the effectiveness of the classifier at detecting DNS tunneling in the wild, a Python script was written to imitate three malware families as described in security blog posts [8] [9] [10]. This resulted in nearly 20,000 queries spanning 400 domains, all of which were correctly classified as malicious by the model. (The test was conducted with a previously trained Random Forest classifier that never saw the malware data.)

IV. Discussion

A. *Individual Domain Analysis*

Some insight can be gained by looking at the behavior of incorrectly classified domains. For example, in the sixth trial, all the supervised models mistook "computerkolkata.com" as malicious and "mal643.com" as benign. The unprocessed DNS queries to these domains are listed below.

computerkolkata.com:
upg9.computerkolkata.com
upg10.computerkolkata.com

mal643.com:
bbnoa.mal643.com
bbnoa.mal643.com
bbboa.mal643.com
bbroa.mal643.com
bbboa.mal643.com
bbfoa.mal643.com

To a human, it seems relatively obvious that the first group of queries is normal and the second is odd (and for reasons besides the presence of the word "mal" in the primary domain – recall that we are only looking at the subdomains). Indeed, "upg9" and "upg10" appear simply to be redundant name servers, which is a common practice in DNS. Meanwhile, the latter queries were generated by a DNS tunneling tool pinging "mal643.com".

Table I. Feature Values for Example Domains

Domain	Feature							
	ENT	***ACR***	***UQV***	***UQR***	***ASL***	***LMW***	***ECR***	***SMR***
computerkolkata.com	0.481	1.0	0.095	1.0	0.086	0.450	0.450	0.667
mal643.com	0.390	1.0	0.181	0.667	0.095	0.567	0.900	0.827

However, consider their feature values as shown in Table I. On most features, "computerkolkata.com" lands in the orange regions of Fig. 3, and "mal643.com" lands in the blue regions. So, it is no surprise that they were classified in this manner. This highlights the need for even more granular approaches, such as sequence detection. The challenge is that each new decision in the model represents both an additional computation cost and a potential vulnerability to be exploited. As usual, a compromise must be reached depending on the application. For cases with a small false positive rate, a simple whitelist is probably suitable.

B. *Effectiveness of the Approach*

The goal of this effort was to develop an analytic capable of detecting DNS tunneling behavior generally. The question must be asked, "What's to prevent an adversary from modifying his behavior to avoid detection?" In short, nothing. If an actor is using the DNS protocol as intended (or appears to in every meaningful sense), then this method cannot identify him. In other words, this analytic is not a blanket fix for DNS security. Instead, it offers visibility into a particular behavior (DNS tunneling) that is a hallmark of malicious activity.

To be clear though, any attempt to evade this analytic would severely cripple client-server transmissions, which is (by definition) what attackers use DNS tunneling for. As an example, suppose an enterprise periodically runs this analytic on the past five hours of DNS traffic. An adversary could limit himself to just a few queries every five hours, but then his data throughput would be somewhere around 1 byte per minute. Or, he could forgo an anti-caching mechanism and send more duplicate queries, but then he could lose packets if they are not forwarded to the server. He could increase the English content in the subdomains, but this would require more or longer queries to transmit the same amount of data. Thus, the attacker is faced with difficult tradeoffs that restrict him to ultra-low throughput applications such as extremely basic command and control.

V. Conclusion

The use of supervised machine learning, enabled by synthesized malicious data from a configurable DNS tunneling tool, greatly improved the precision and recall of DNS tunneling classifiers as compared with unsupervised learning. Keeping in mind the tradeoff between performance and training time, one of the estimators discussed can be used as an advanced analytic to detect DNS tunneling in query traffic. The success of the approach taken affirms the recommendation of Nadler et al. to attempt to classify malicious primary domains instead of individual queries. It also establishes the effectiveness of the statistical features

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

discussed in this paper at distinguishing DNS tunneling behavior from normal usage.

This work leaves open the exploration of more powerful machine learning methods, such as neural networks. There also may be other features or sources of data that would be useful in this effort. For example, the contents of the query response could be an additional vector for determining the amount of data being transmitted between a client and server. This may enable detection of even the most discrete activities, such as ultra-low throughput command and control.

ACKNOWLEDGMENT

The author would like Randy Charland, Kyle Nolan, and Kevin Grace for guidance and support on the project, and Dr. Karen Panetta for academic mentorship.

REFERENCES

[1] S. Ariyapperuma and C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC," in *The Second International Conference on Availability, Reliability and Security (ARES'07)*, Vienna, 2007.

[2] EfficientIP, "Global DNS Threat Report," 2018. Available: https://www.efficientip.com/resources/dns-security-survey-2018/. [Accessed 2 May 2019]

[3] M. Aiello, M. Mongelli and G. Papaleo, "Basic classifiers for DNS tunneling detection," in *2013 IEEE Symposium on Computers and Communications (ISCC)*, 2013.

[4] A. Nadler, A. Aminov and A. Shabtai, "Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol," arXiv.org, 2018.

[5] USC/B-Root Operations with USC/LANDER project, *Day In the Life of The Internet (DITL),* 2018.

[6] P. Mockapetris, "Domain Names - Implementation and Specification," November 1987. [Online]. Available: https://tools.ietf.org/html/rfc1035. [Accessed 14 May 2019].

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos and D. Cournapeau, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, p. 2825–2830, 2011.

[8] G DATA Blog, "New FrameworkPOS variant exfiltrates data via DNS requests," G DATA, 15 October 2014. [Online]. Available: https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests. [Accessed 14 May 2019].

[9] A. Shulmin and S. Yunakovsky, "Use of DNS Tunneling for C&C Communications," Kaspersky Lab, 28 April 2017. [Online]. Available: https://securelist.com/use-of-dns-tunneling-for-cc-communications/78203/. [Accessed 14 May 2019].

[10] S. Yunakovsky and I. Pomerantsev, "Denis and Co.," Kaspersky Lab, 25 January 2018. [Online]. Available: https://securelist.com/denis-and-company/83671/. [Accessed 14 May 2019].