Chapter 40
Detecting Domain Name System Tunneling and Exfiltration from Domain Name System Traffic

Yi-Chung Tseng, Ming-Kung Sun, and Wei-An Chen

Abstract In this study, we present a framework to detect a Domain Name System (DNS) tunnel and DNS exfiltration network traffic by using both unsupervised and supervised learning algorithms. In general, considerable time is required to learn the structure of the data before clustering when using an unsupervised learning algorithm. Therefore, in this study, we leveraged the power of mathematical algorithms for calculating the optimal number of clusters and reducing the time required for understanding the data structure. Conversely, we used a supervised learning method to learn the data leakage behavior for detecting DNS exfiltration traffic. We used an open-source tool to generate testing data, and the experimental result proved the robustness of the proposed framework.

40.1 Introduction

The DNS [1] is an Internet service. A domain name is a characteristic structure that is easier to understand and remember than an Internet Protocol (IP) address. The DNS acts as a decentralized database that maps a domain name to an IP address. In addition to being used for providing Web services, Web hosts, and other online services, domain names are often used by attackers to steal personal information. For example, an attacker can attach personal information encryption to the domain name itself, such as "ojswczdnmuxg2zd4ge3q.malware.com," through the Base64 encryption algorithm. The aforementioned domain name can be split into three blocks. The first block is "ox-wczdnmuxg2zd4ge3q;" the second block is ".malware;" and the third block is ".com."

Y.-C. Tseng (✉) · M.-K. Sun · W.-A. Chen
Acer Cyber Security Incorporation, Taipei, Taiwan
e-mail: Bruce.Tseng@acercsi.com

M.-K. Sun
e-mail: Morgan.Sun@acercsi.com

W.-A. Chen
e-mail: Wayne.Chen@acercsi.com

The string in the first block represents the personal data encryption function hidden in the domain name itself by the attacker through an encryption algorithm. The second and third blocks specify the actual DNS server of the attacker. When an attacker desires to steal information, such as personal data which can be encrypted into the subdomain name, consequently, the domain name along with the target data is transmitted from the client server to an external DNS server through the DNS recursive technology, thus causing data leakage.

About the recursive DNS technique used for data leakage, a client sends a query to the local DNS server. If the server does not respond, the local DNS server continues to query the higher layers of the DNS server until the IP with the answer is found. The process is initiated from the root directory. For example, for www.google.com.tw, the ".tw" query is executed first. Then, ".com" is executed from the top level to the second level and so on.

Another method of conducting data leakage is by establishing a DNS tunnel. A hacker can use the tunnel to transmit the target data. This study proposes an architecture that can detect DNS tunneling and DNS exfiltration network traffic by leveraging the power of a modern machine learning algorithm. The experimental results indicate the simplicity, robustness, and scalability of the proposed approach.

The remainder of this paper is organized as follows. Section 2 introduces the relevant literature related to DNS tunneling and exfiltration. Section 3 presents an overview and the framework of the approach. Section 4 demonstrates the experimental results. Section 5 summarizes the conclusions of this study.

40.2 Related Work

40.2.1 DNS Tunnel

Establishment of a DNS Tunnel Anirban et al. [2] mentioned that when a client tries to establish a DNS tunnel with an external DNS server, the query that contains the TXT record is sent, which is a type of resource record in DNS. Once the server receives the request, it returns a response with the TXT records to the client. Thus, a DNS tunnel is established successfully. A malicious server can take advantage of this tunnel to establish a tunnel for starting a session or executing an instruction.

Detection of a DNS Tunnel Anirban et al. [2] used the *k*-means clustering algorithm to find a DNS tunnel. Binsalleeh et al. [3] characterize the malicious payload distribution tunnel in DNS. They proposed solution characterizes these tunnels based on the DNS query and response messages patterns. Farnham and Atlasis [4] presented an overview of the history and techniques used for DNS tunneling detection. Compared with the regular A or AAAA DNS queries that have a constant size range, tunneling traffic tends to have a considerably larger size range. Paxson et al. [5] use the implementation of Kolmogorov complexity to detect DNS tunnel. Dietrich et al. [6] used various features to cluster DNS traffic by using *k*-means clustering with $k = 2$. Born [7] proposed a method to detect DNS tunnels by using meaningful words. Hind [8] proposed a neural network for detecting DNS tunnel but did not provide any information on the data sources, features used, model selected, or model performance.

40.2.2 DNS Exfiltration

Situational explanation of DNS Exfiltration Anirban et al. [2] mentioned that assumed a scenario in which a computer is infected by a malicious program that wants to steal and transmit information to the attacker server. In the first step, the malware encrypts the private data. In the second step, the encrypted data is attached to the attackers' domain. For example, if aGVs13IGb3Vu is the encrypted data and malware attaches this data to.malware.com, the domain name aGVs13IGb3Vu.malware.com is obtained. In the third step, because the encoded domain is not part of the local cache, the domain is forwarded to the server of malware.com by using the DNS recursive technology. Once the attackers' DNS server receives the query, the attacker can extract the third-level domain and decode it. In the fourth step, the attacker can respond to the client, which appears benign. The following sections describe how DNS exfiltration can be detected using the proposed method.

Signature-based Detection of DNS Exfiltration Jawad et al. [9] mentioned traditional methods for detecting DNS exfiltration rely on signatures which are not sufficient. By registering a new domain name, an attacker can easily bypass the blacklist. Besides, the signature-based approach relies on rules checking and thresholds to trigger an alert and is struggling to discover the malware's pattern behavior. On the other hand, maintaining a blacklist is also inefficient.

Rule-based Detection of DNS Exfiltration Fawcett [10] described several encoding techniques for DNS exfiltration. These techniques rely on rule-based detection, such as detection according to the number of requests and responses, entropy of the hostname, percentage of numbers in the domain name, and number of non-existed domain.

Machine Learning-based Detection of DNS Exfiltration Anirban et al. [2] proposed machine learning models, by using logistic regression model to predict DNS exfiltration; they used eight features to describe the domain string and exfiltration domains as a negative set and benign domain as a positive set to train the model.

40.2.3 Unsupervised Learning

***k*-Means Clustering** One of the most well-known unsupervised methods is the k-means clustering algorithm. The user randomly selects k points as the initial centroid, where k is the user-specified parameter. Each point is then assigned to the cluster with the closest centroid. The centroid of each cluster is then updated by calculating the average of the data points for each cluster. The centroid is repeatedly assigned and updated until no improvement is obtained by changing the cluster or until the centroids are the same again.

Silhouette Method Rousseeuw [11] proposed the silhouette method, which is used for interpreting and verifying consistency within a data cluster. This technique provides a concise graphical representation of the classification of each object.

40.2.4 Supervised Learning

Extreme Gradient Boosting (XGBoost) XGBoost [12] is an optimized distributed gradient boosting library that is designed to be highly efficient, flexible, and portable. This library employs machine learning algorithms under the gradient boosting framework. XGBoost provides parallel tree boosting, which solves many data science problems rapidly and accurately.

40.3 Problem Formulation

We assumed a scenario in which an attacker hacks a client computer and installs a backdoor program. When attackers desire to send a command or steal personal information, they can use a DNS tunnel to establish a connection. Once a DNS tunnel is established, the encrypted private data is sent out through the DNS tunnel. The attackers can also use recursive DNS technology to send out the exfiltration data. Hence, we propose a detection method that leverages a machine learning algorithm to identify DNS tunneling and DNS exfiltration. Unsupervised and supervised learning are used in the proposed method. The proposed approach is introduced in the following section.

40.4 Overview of the Approach

The proposed architecture relies on analyzing DNS traffic and can identify DNS tunneling and DNS exfiltration. The proposed process is explained briefly in the following text. First, DNS traffic is collected. Second, feature engineering technology is used to extract features from the DNS query, including TXT and A records. Third, the silhouette method is used to calculate the optimal number of clusters with the k-means clustering algorithm by using the aforementioned features for determining

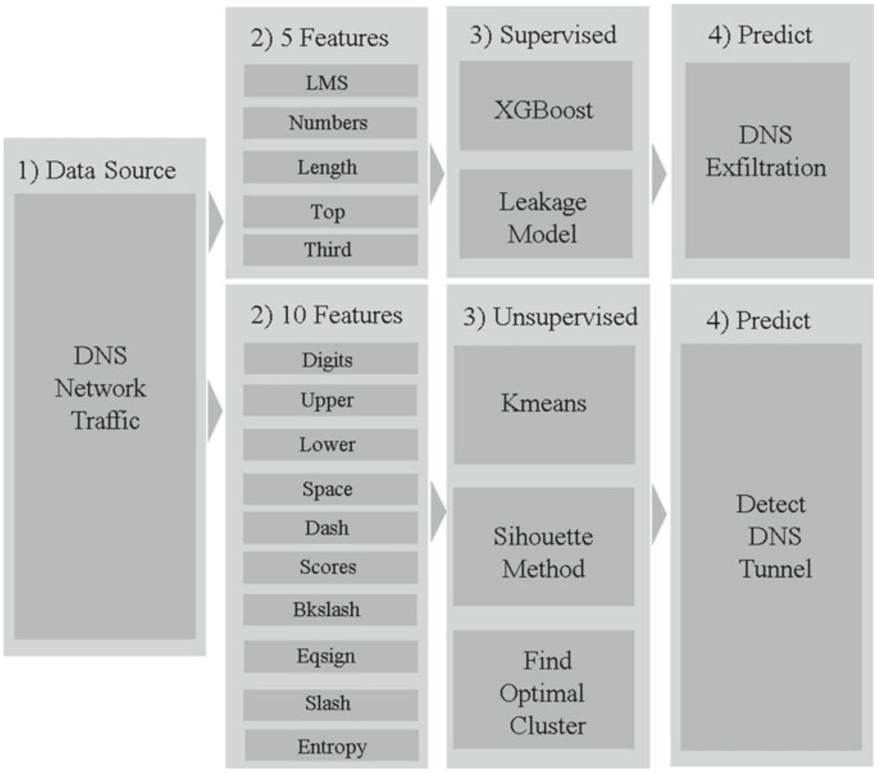

Fig. 40.1 Overview of the proposed approach for detecting DNS tunneling and DNS exfiltration

whether a DNS tunnel exists. We also trained a model that can identify exfiltration from a DNS query, as presented in Fig. 40.1.

40.4.1 DNS Query Collection

In the first step, the DNS network traffic is collected. Because there exist many types of data in DNS traffic, such as A, AAAA, PTR, and TXT, we only collect TXT, A, and AAAA records from DNS traffic.

40.4.2 Feature Engineering

In the second step, clustering is conducted on the TXT records by using feature engineering. Anirban et al. [2] used 12 features to describe the behavior of TXT

Table 40.1 Feature engineering for detecting DNS tunneling

Feature	Explanation
Number of digits	Calculate how many digits exist in the TXT record
Number of upper	Calculate how many upper cases exist in the TXT record
Number of lower	Calculate how many lower cases exist in the TXT record
Number of space	Calculate how many spaces exist in the TXT record
Number of dash	Calculate how many dash exist in the TXT records
Number of under line	Calculate how many under lines exist in TXT records
Number of slashes	Calculate how many slash exist in TXT records
Number of back-slash	Calculate how many back-slash exists in TXT records
Number of equal-sign	Calculate how many equal-signs exist in TXT records
Entropy	Calculate the Shannon entropy of TXT record

Table 40.2 Feature engineering for DNS exfiltration

Feature	Explanation
all_domain_length	Calculate the length of the entire domain name
third_domain_length	Calculate the length of the 3LD
domain_num_percentage	Calculate the proportion of numbers in the domain name
domain_LMS_percentage	Calculate the ratio of the longest and most meaningful string in the domain name to the overall string
top_domain_count	Calculate the same 1LD domain name as the total number of domain names

data. Therefore, in the present study, the 10 features presented in Fig. 40.1 were used. The meaning of each feature is provided in Table 40.1.

To train a model that can identify DNS exfiltration, feature engineering is required. We used five features to describe the behavior of a DNS query string. The meaning of each feature is specified in Table 40.2.

40.4.3 Determining the Optimal Number of Clusters that Can Find a DNS Tunnel

In the third step, an unsupervised algorithm is used to cluster the TXT records. Because the TXT records of DNS traffic differ with the company environment, the clustering value should be dynamically adjusted according to the DNS traffic. Therefore, the proposed work used the silhouette method to proactively find the optimal number of clusters. After clustering the data, we can observe whether each cluster has encoded strings. If there are encoded strings in a cluster, it can be concluded that there is a DNS tunnel in the DNS network.

40.4.4 Training an XGBoost Model for Identifying DNS Exfiltration

In the fourth step, we must train a model to identify DNS exfiltration by using a supervised learning algorithm. The preparation of a supervised model requires a labeled dataset. Therefore, we used a dataset that has only one field and contained a large number of leakage domains. This dataset [13] was provided by the Sydney University. Moreover, we used a credit card generator to generate a large number of credit card numbers; encrypt the card numbers through md5, base64, and other encryption algorithms; and then attach the encryption string into a domain. We also used a domain that was not leaked and was provided by Alexa [14]. The aforementioned datasets were integrated into a single dataset, including the leaked and benign domain datasets. Then, five features were used to characterize the behavior of the leaked and not leaked domains, as presented in Table 40.2. Finally, we used the XGBoost algorithm, which is a supervised learning model, to learn from the dataset for identifying DNS exfiltration.

40.5 Experiment

Detecting DNS tunneling by examining TXT records is a difficult problem primarily due to the high diversity of TXT records in real-world DNS traffic. Hence, we used a testing dataset obtained from an open source for detecting DNS tunneling. Moreover, we used the Data Exfiltration Toolkit (DET) [15] to generate a large number of DNS exfiltration samples for verifying whether our model could identify DNS exfiltration.

40.5.1 DNS Tunnel Detection

Data Collection The testing dataset [16] for DNS tunneling contained 1096 TXT records, which accounted for approximately 0.054308% of all DNS queries. Some TXT records were generated by incorporating DNScat [17] tunneling traffic, and the other records were regular DNS traffic.

***k*-Means Algorithm and Average Silhouette Method** We selected *k*-means clustering by using the aforementioned features to detect all the TXT queries that are encoded a string. The optimal number of clusters was calculated using the average silhouette method, as displayed in Fig. 40.2. In the figure, the *x*-axis represents the number of clusters and the *y*-axis represents the silhouette score. The higher the *y*-axis score, the better is the clustering result. In this study, the highest silhouette value was obtained by dividing the datasets into three clusters.

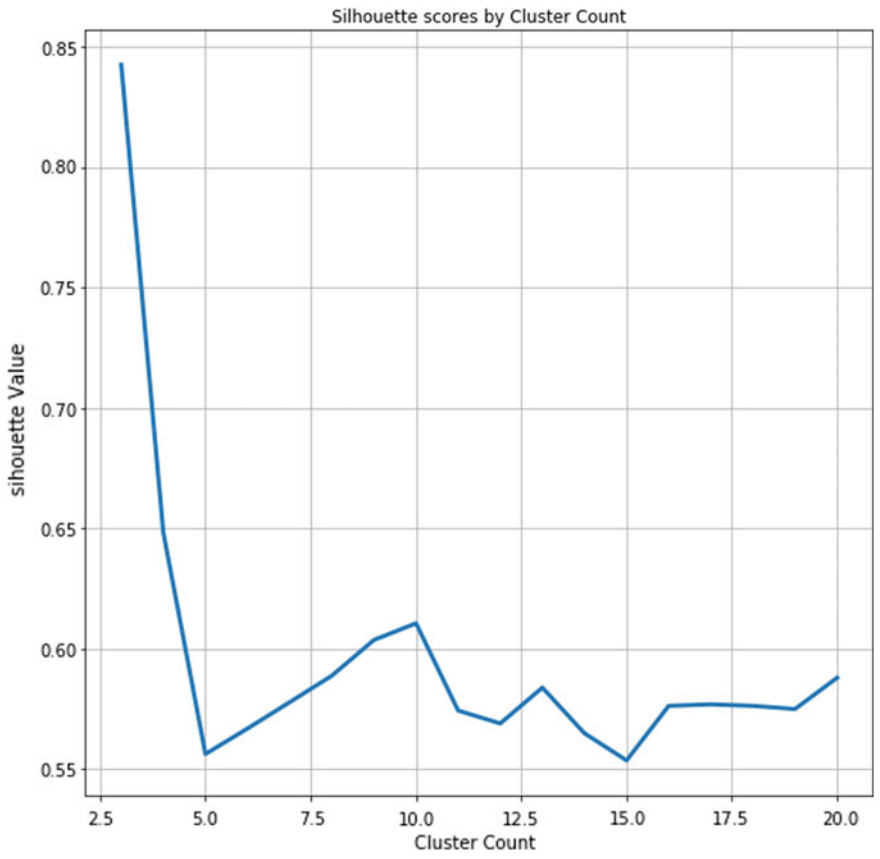

Fig. 40.2 Sihouette scores by cluster count

Cluster and TXT records As shown in Table 40.3, we decided that clusters 1 and 2 contained encoded TXT responses and that cluster 0 contained site verification traffic. Although clusters 1 and 2 provided the same type of encoded TXT record, the length of the encoded string in cluster 1 was longer than that in cluster 2. Moreover, a small number of encoded strings were classified into cluster 0. Overall, from the clustering results, we can quickly identify if a DNS tunnel exists.

40.5.2 DNS Exfiltration Detection

Data Collection In this experiment, we used a credit card generator [18] to generate a large number of credit card numbers, encrypted the card numbers through md5, base64, and other encryption algorithms; and then combined the data into a domain.

Table 40.3 Cluster of TXT records

Cluster	TXT records
0	254.229.168.192.spam.dnsbl.sorbs.net VERSION.BIND version.bind 251.229.168.192.spam.dnsbl.sorbs.net cf._dns-sd._udp.0.95.168.192.in-addr.arpa dnscat.690001a18087f230aaa1840010aecff573
1	dnscat.4fc801a1803d607cb283df007ea280e2d578be0e25c47557d1f4db0d3faa.5d36c542b7b8248cf30b9c2c488b1669d80338f4849541 ad7d4f96ea5b76.5f4e43420d69c7ec8e7c2bdbc15068bc9117a4f69194f780e2a68b477d8f.e6fd112c94374bc35c3e586ed675dde5cbb7c058ad5bbe9fbc dnscat.c68801a180bd2c0a8262b000a2133c316f63766097374ef8fe820302d52e.4083f9058db6661c4c045f3e1a9d62830430c7 cf4b98e64b0fac3aabd38e.e142207b5b81c6254cda8cc3343167e837e938d67dc45c2f8b2c6a2222c6.9c6ce247511c002050c3a7bfc11f9205f12c9ed7b4d96dbe1a
2	dnscat.0ad001a1807f5322891563000c1038c3c7a525f6d975b68f3d73df039250.3db23e96eee3b7a4c2a261d88966c6491a731b3f1f32fd1b3039f6270088.f5c730 dnscat.137903a180000000008187ab6e71ed60866c8647ed9ede53317c6ca4bca9.9f8756ac08ba26d904ab5ddcb39f3ff291d5c 5e50ff3287dd350a7a6c4eb.58eff3d0c8b589dea47c563e1b

Table 40.4 Credit card number encryption

Credit card number	Encryption
4916658665745840	Mzc2MzgwNTA3NTY3NzAx5
30360128837301	30993683d51e835756d02f655af05ac
30368674657247	1e8299e0c7a1690ec3d6928f2a8366a
4916658665745840	275c15e507e168f5f71eb848cd56cfd3180c0a33
4556229341515026	275c15e507e168f5f71eb848ctq6cfd3180ec0a33

Table 40.4 lists the original credit card number and the corresponding base64 encryption data. Table 40.5 presents the domain name with the encrypted data. We combined 1,435,514 exfiltration domains as a negative set and 1,000,134 benign domains from Alexa as a positive set to train the model.

XGBoost Our feature space had five dimensions, as listed in Table 40.2. We selected the XGBoost algorithm because this algorithm is easy to deploy, effective, and exhibits superior performance. We used Scikit-learn [19] to deploy XGBoost and to validate and test our proposed method.

Self-verification For testing the effectiveness of the XGBoost model, we validated the metric Roc_auc, which is a performance measure, at various threshold settings (Table 40.6). Roc is the probability curve, and auc represents the degree or measure of separability. The precision score is obtained by calculating the ratio of all “correctly retrieved results (TP)” to all “actually retrieved (TP + FP).”

Effectiveness of the Model To demonstrate the effectiveness of the proposed model, we used the DET to generate a leakage domain along with the regular domain. As displayed in Table 40.7, five malicious domains were predicted using the developed model. The output value is a probability value. If the probability is more significant than 0.5, the domain is considered to have exfiltration. For

Table 40.5 Encrypted data attached to the domain name

Encryption	Domain name
Mzc2MzgwNTA3NTY3NzAx5	Mzc2MzgwNTA3NTY3NzAx5.malware.com
30993683d51e835756d02f655af05ac	30993683d51e835756d02f655af05ac.malware.com
1e8299e0c7a1690ec3d6928f2a8366a	1e8299e0c7a1690ec3d6928f2a8366a.malwarea.com
275c15e507e168f5f71eb848ctq6cfd3180ec	275c15e507e168f5f71eb848ctq6cfd3180ec0a33.malware.com

Table 40.6 Validation metrics

Metric	Our proposed method
roc_auc	1.0
precision	0.9999992885195514

Table 40.7 Domain and probability

Domain	Probability
59-124-10-43.hinet-ip.hinet.net	0.000015615
59-124-106-74.hinet-ip.hinet.net	0.0000823
130353161663731383536663564646236.google.com	0.63
init.ojswczdnmuxg2zd4ge3q.base64.systw.net	0.56

example, for init.ojswczdnmuxg2zd4ge3q.base64, the probability value corresponding to.systw.net was 0.56. Thus, this case is considered to have exfiltration. Conversely, the probability value corresponding to 59-124-106-74.hinet-ip.hinet.net was 0.00008032. Thus, this case does not have exfiltration.

40.6 Conclusion

In this study, we present a framework to detect DNS tunneling and DNS exfiltration through the DNS network traffic. This framework uses unsupervised learning and the silhouette method to determine the optimal number of clusters for identifying DNS tunneling. We also used a supervised learning algorithm to train the XGBoost model for identifying whether any information leakage occurred in the traffic. Finally, to verify the effectiveness of our architecture, we used open-source DNS network traffic that contained tunneling and exfiltration. The proposed framework can accurately detect tunneling and exfiltration and prove the robustness, simplicity, and scalability of our method.

References

1. DNS: https://en.wikipedia.org/wiki/Domain_Name_System
2. Anirban, D., Min-Yi, S., Madhu, S.: Detection of exfiltration and tunneling over DNS. In: 16th IEEE International Conference on Machine Learning and Applications (2017)
3. Binsalleeh, H., Kara, A.M., Madhu, S., Debbabi, M.: DNS noise: characterization of covert channels in DNS. In: Mobility and Security (NTMS) (2014)
4. Farnham, G., Atlasis, A.: Detecting DNS Tunneling. SANS Institute InfoSec Reading Room (2014)
5. Paxson, V., Christodorescu, M., Javed, M., Rao, J.R., Sailer, R., Schales, D.L., Stoecklin, M.P., Thomas, K., Venema, W., Weaver, N.: Practical comprehensive bounds on surreptitious communication over DNS. In: USENIX Security (2013)
6. Dietrich, C.J., Rossow, C., Freiling, F.C., Bos, H., Van Steen, M., Pohlmann, N.: On bot-nets that use DNS for command and control (2011)
7. Born, K.: DNS tunnel detection using character frequency analysis (2010)
8. Hind, J.: ExFILD: Catching DNS tunnels with ai (2009)

9. Jawad, A., Hassan, H., Qasim, R., Craig, R, Vijay, S., Lee.: Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks. Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia (2019)
10. Fawcett, T.: ExFILD: a tool for the detection of data exfiltration using entropy and encryption characteristics of network traffic (2010)
11. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Comput Appl Math (1987)
12. XGBoost: A Scalable Tree Boosting System
13. Datasets: http://downloads.majestic.com/majestic_million.csv
14. Alexa: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
15. Data Exfiltration Toolkit: https://github.com/qasimraz/DET
16. TXT Record: https://github.com/chuayupeng/dns-tunnelling-detection/tree/master/pcap
17. DNScat: http://bit.ly/1PhF8Qd (2004)
18. Credit card generator: https://www.fakepersongenerator.com/credit-card-generator
19. Scikit-learn: https://scikit-learn.org/stable/