Detecting DNS Tunnel through Binary-Classification Based on Behavior Feature

Jingkun Liu[*†], Shuhao Li(Corresponding author)[*], Yongzheng Zhang[*], Jun Xiao[‡] and Chengwei Peng[†§]
[*]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[†]School of Cyber Security, University of Chinese Academy of Sciences
[‡]Huawei Technologies Co., Ltd
[§]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Email: xiaojun23@huawei.com,{liujingkun,lishuhao, zhangyongzheng, pengchengwei}@iie.ac.cn

***Abstract*—DNS tunnel is a typical Internet covert channel used by attackers or bots to evade the malicious activities detection. The stolen information is encoded and encapsulated into the DNS packets to transfer. Since DNS traffic is common, most of the firewalls directly allow it to pass and IDS does not trigger an alarm with it. The popular signature-based detection methods and threshold-based methods are not flexible and make high false alarms. The approaches based on characters distribution features also do not perform well, because attackers can modify the encoding method to disturb the characters distributions.**

In this paper, we propose an effective and applicable DNS tunnel detection mechanism. The prototype system is deployed at the Recursive DNS for tunnel identification. We use four kinds of features including time-interval features, request packet size features, record type features and subdomain entropy features. We evaluate the performance of our proposal with Support Vector Machine, Decision Tree and Logistical Regression. The experiments show that the method can achieve high detection accuracy of 99.96%.

I. Introduction

The covert channel detection techniques have received much attention for the reason that the covert channels have become sophisticated means to transferring information. DNS tunnel is a typical covert channel using DNS protocol to encapsulate traffic. Domain Name System (DNS) [1] plays a critical role in the operation of the Internet nowadays and provides the services of two-way mapping between domain names and IP addresses. Currently, network security devices do not pay attention to the DNS tunnel. Most of the firewalls do not block DNS packets and an IDS does not raise an alarm when a DNS packet seems normal (with normal DNS packet format). Thus, DNS tunnel is attractive to attackers and is widely used to transfer the stolen information [2]. There have been several open-source DNS tunnels tools, such as Iodine [3], dns2tcp [4], DNScat2 [5]. And the popular penetrate tool Metasploit provides DNS tunnel module [6].

In recent years, some researches have been studied to detect DNS tunnels. One kind is the traditional signature-based method [7]. Attackers defenders analyze the tunnel traffic and extract the tunnel signatures. IDS with these signatures can accurately detect DNS tunnels. Another kind is the machine learning methods [8]. By analyzing the traffic difference between the normal DNS traffic and tunnel traffic, attacker defender can extract the tunnel features to train a classifier. Then the classifier can be used to distinguish tunnel traffic with the normals.

Although many approaches have been proposed to identify the DNS tunnel, some issues still deserve further attention. The signature-based mechanism can only be applied to traffic encapsulated by known tunnel tools. The presumption of machine-learning methods is that the character distribution of tunneled traffic is different with normal. However, the character distribution of tunneled traffic generated by some tools is similar to the normal DNS queries. Thus, this kind of method is not suitable for all kinds of tunnel tools.

In order to address the issues above, we propose an effective and applicable DNS tunnel detection mechanism. Our approach can analyze the DNS traffic of Recursive DNS (RDNS) for detection. Similar to the machine-learning method discussed above, the key component our mechanism is also a classifier. The main difference between our method and them is that we not only focus on the character distribution differences between normal and tunnel traffic but also time frequency, DNS record types and DNS query length. Accordingly, we use 4 kinds of features (totally, 18 features), including the time-interval features, request packet size features, domain entropy features and record type features. Our method can detect kinds of DNS tunnels with high accuracy.

We validate the performance of our proposal with real DNS traffic. The result shows that our method can detect DNS tunnels in time. After analyzing 15 DNS query packet, the detection accuracy can reach 99.96% . Our method can achieve high performance with different kinds of classifiers, including Support Vector Machine, Logistical Regression, and Decision Tree.

In summary, we make the following contributions:

- We propose an effective and applicable DNS tunnel detection mechanism. It can accurately identify various kinds of DNS tunnels traffic produced by available tunnel tools.
- We deeply analyze the behavior of DNS tunnel traffic and extract 4 kinds of features (18 behavioral features). The features can be helpful to build robust tunnel detection system.
- We conduct experiments on real-world DNS traffic to evaluate the performance of our proposal. The results

show that our approach can detect DNS tunnel accurately (99.96% detection accuracy, 99.98% precision and 99.93% recall).

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the mechanism of DNS tunnel and presents the analysis of the DNS tunnel. We present the overview of our approach in Section 5 and deeply analyze 4 kinds of features we use in the identification of DNS tunnel in Section 4. The experiments we conducted on real-world dataset are described in Section 6. Finally, Section 7 concludes the paper.

II. Related Work

To the best of our knowledge, there are two main types of DNS tunnel [9]: time-interval based DNS tunnels and storage based DNS tunnels [10]. The former one uses timing properties of DNS requests to transmit messages to the C&C and the latter one overloads the protocol to encapsulate encoded information. The time-interval based DNS tunnels are very stealthy, however, the capacity of time-interval based DNS tunnels are so small that the related utilities are very few. The latter one can provide higher bandwidth and better reliability compared with time-interval based tunnels, so the researchers focus on detecting storage based DNS tunnels. In reality, many of the DNS tunneling implements do not try to be stealthy. They are relying on the fact that DNS traffic is often not monitored by security device and policy.

There are various related research efforts that attempt to address the problem of DNS tunnel and various DNS detection approaches have been proposed.In general, DNS tunnel detection follows two lines of research: The first line of research, called payload analysis, tries to detect malicious DNS traffic through the features of one or more requests and responses. The second line of research focusing on the features of traffic over time is called traffic analysis.

A. Payload Analysis Methods

In past research, various threshold-based techniques have been used to identify DNS tunnel packets. Firstly, Bianco defines methods to distinguish suspicious DNS tunnel packet from normal traffic based on the ratio of the size of the request and response in a blog post [11]. An alarm is raised if a certain threshold is exceeded. In addition, looking for records that are not commonly used by a client such as TXT records is another possible detection method [12]. These abnormal detection methods based on threshold are not flexible enough causing high false positive rate or false negative rate.

Except for features above, researches on Domain Generation Algorithms (DGA) [13] provide other payload analysis detection techniques. These methods focus on randomness detection of domain mainly. DGA generated domains are abnormal in a similar way to names from data encoding. The malicious domains and subdomains used by covert channel have more randomness in general and the more randomness in the string creates a higher the entropy [14] according to the formula developed by Claude Shannon. So the researchers proposed detection approaches based on the entropy of requested hostnames [15], [16]. After that, Kenton Born and Dr. David Gustafson propose using character frequency and N-gram analysis to detect covert channels [17]. Cheng Qi proposes a scoring mechanism that based on bigram character frequency which can distinguish normal domain names and random ones. [18]. However, the scheme above can not resolve the problem of high false negative that subdomains of the DNS tunnel background traffic such as 'nxcviC+gBA' have similar Shannon Entropy to the complex normal domain.

There are also a group of approaches based on specific signature [7] for popular detection platforms like snort. The static nature of the signatures means that they are not effective enough to identify more than a small portion of the available tunneling tools.

B. Traffic Analysis Methods

Traffic analysis involves looking at various request/response pairs over time. Existing traffic analysis methods detect DNS tunnels by the flow (a set of packets passing an observation point in the network during a certain time interval and having a set of common properties)features such as volume of DNS traffic per IP address, volume of DNS traffic per domain [19], number of hostnames per domain [20]. For example, Wendy Ellens and Anna Sperotto have proposed a detection method based on flow data [21]. These approaches ignore the observation may be at Recursive DNS server which could receive enormous number of DNS packets. In addition, the feature of flow data needs a period of time to detect so it can not give the alarm in real time.

Except for solutions above, Maurizio Aiello proposed the supervised learning approaches with majority voting for DNS tunnel detection. The main difference from ours is that their approach works on the overall exchange data [8] while our approach focuses on a set of packets for a specific domain.

III. DNS Tunnel Analysis

When an application makes a request that requires a domain name look up, the network host firstly checks to see if it has a DNS record associated with the domain name in the local cache which contains the recent lookups. If the local cache can provide the answer to the request, the value in the cache is returned. Otherwise, the request is sent to Recursive DNS server. To improve efficiency, reduce DNS traffic across the Internet, RDNS server cache DNS query results for a period of time (time-to-live). If RDNS server does not have the DNS record cached, it begins the recursive process of going through the Authoritative DNS hierarchy. Authoritative DNS servers are responsible for providing responses to RDNS server which contains important information for each domain, such as corresponding IP address.

A. DNS Tunnel Mechanism

The mechanism of DNS tunnel is shown in Figure 1. There is a malicious host inside the organization and DNS packets might be allowed while most other types of channels

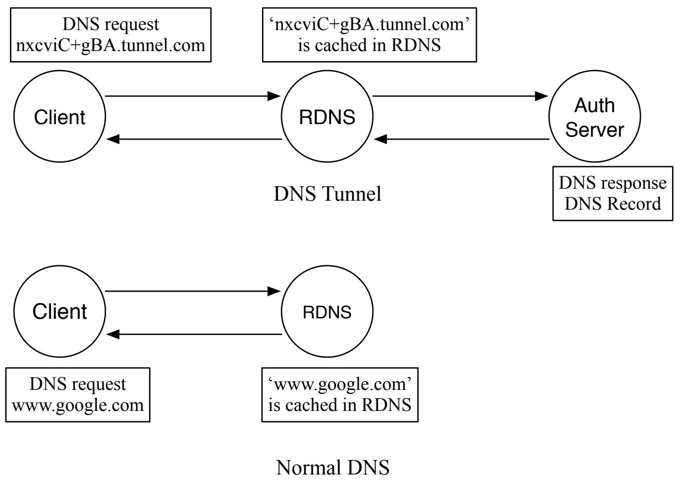


Fig. 1: DNS tunnel mechanism

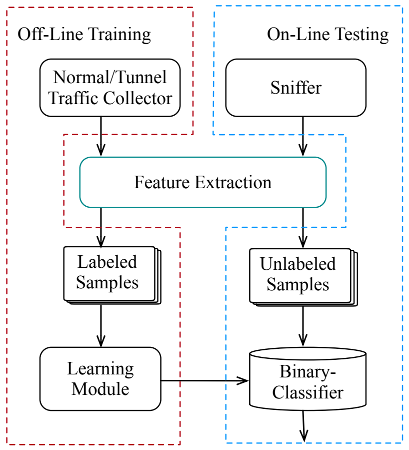


Fig. 2: Architecture of Our Approach

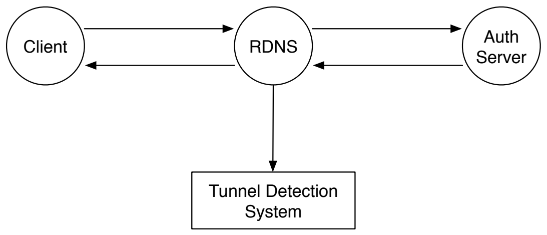


Fig. 3: Deployment

are blocked or filtered by firewalls or IPS. Firstly, the DNS tunnel application which runs on the malicious host makes a request for a domain name lookup. The subdomain of an intended domain name is the encoded information and is not duplicate over a period of time (TTL). Then the host sends the query request to RDNS and RDNS can not find the domain name in the cache. Finally, the request is forwarded to the Authoritative name server that is DNS tunnel server host. The tools on the DNS tunnel server host decodes the information encapsulated in the request packet. After processing, the server encodes the response data and encapsulate it into answer packet. Eventually, the response is sent back to the requesting host.

All of the utilities use similar techniques but have variations on encoding, record types and other implementation details. For example, the encoding technique includes base32, base64, base64u and others. Different implement details lead to different performance. Tom van Leijenhorst investigates the viability and performance of DNS tunnel [22]. The core components used by all tools contains an Authoritative DNS server with a public IP address, a controlled domain which is used to define the Authoritative DNS server for that domain, and the compromised client host. Once the server host and client host are ready to transfer data, they will work as described above.

B. Related Implements

There is a list [23] of DNS tunnel tools in SANS Institute InfoSec Reading Room. However, some of them are just demos [24] and the authors even do not maintain it now, such as tcp-over-dns [25]. Fortunately, there are still several usable DNS tunnel tools for us to analyze: 1) dns2tcp [4] 2) dnscat2 [5] 3) iodine [3] 4)Ozymandns [26]. All of the tools are client-server software. Dns2tcp and Ozymandns are simpler than other tools, but they are still workable. Iodine is a typical tool that tunnels IPV4 data through a DNS server and it can run on Linux, Windows, Mac OS X and other platforms. Dnscat2 is designed to create an encrypted command-and-control (C&C) channel over the DNS protocol and this is also the main difference between Dnscat2 and other DNS tunnel tools. All of them provide various techniques such as data encoding, DNS record type, sizes of fragments for users.

IV. Overview of the Approach

In this section, we introduce our effective and applicable mechanism. The overview of the architecture is shown in Figure 2.

A. Architecture

Our approach consists of four components. The first component, the Sniffer, records the DNS traffic produced by the monitored network in the On-Line Detecting mode. The second component, the Normal/Tunnel Traffic Collector, stores the Off-Line Training traffic. Both of them are responsible for collecting DNS traffic. The collected traffic will be fed into next model, Feature Extractor. The main difference between them is that the traffic from former one will be determined whether it is tunnel traffic while the traffic from latter one is known to be normal DSN traffic or tunnel traffic.

The third component is Feature Extractor. It is responsible for extracting features from collected traffic in the last stage and transforming the extracted features into a feature vector which has been described in Section V in an automatic manner.

The output of Feature Extraction from Normal/Tunnel Traffic Collector is labeled data. we use these labeled data as our training set. Let $\{(x^k, y^k)$, k=1,...,N$\}$ be the training set (N is the size of training set), where x^k is a realization of a feature vector and y^k belongs to $\{0,1\}$. If the features contained in x^k

TABLE I: Features

Category of Features	Description of Features	Number of Features
Time-Interval	Mean and variance of time-interval between a request and response	2
Request Packet Size	Mean and variance of request packet size	2
Subdomain Entropy	Mean and variance of unigram entropy	6
	Mean and variance of bigram entropy	
	Mean and variance of trigram entropy	
Record Type	Frequency of A resource record	8
	Frequency of AAAA resource record	
	Frequency of TXT resource record	
	Frequency of NULL resource record	
	Frequency of CNAME resource record	
	Frequency of MX resource record	
	Frequency of NS resource record	
	Frequency of PTR resource record	
	Total	**18**

corresponds to a DNS tunnel traffic, y^k = 1, y^k=0, otherwise. The training set is fed into the fourth component: Learning Module. This module makes use of the training set to build DNS tunnel detection model. Consequently, these models, and the unlabeled dataset from Sniffer, become an input to the fifth component: The Binary-Classifier.

The final component is the binary-classification model which takes decisions according to the detection models produced by the Learning Module component so that the unlabeled data are grouped into two classes: DNS tunnel traffic and normal DNS traffic.

B. Deployment

Our approach includes two steps: off-line training and on-line detection. In the first phase, we train the classifier with the DNS tunnel and normal traffic. In the second phase, the classifier identifies the tunnel traffic. The deployment of our system is shown in Figure 3, the system sniffs the traffic of the Recursive DNS.

V. Feature Selection

To determine the DNS traffic features that are indicative of tunnel behavior, we studied traffic of an ISP DNS server and Intranet DNS server respectively. The Intranet DNS server provides domain name lookup service for Intranet client hosts. Four kinds of DNS tunnel tools including dns2tcp [4], dnscat2 [5], iodine [3] and Ozymandns [26] run on Intranet client hosts. After the analysis period, we identified 4 kinds of features that are able to characterize DNS tunnel traffic. Table I gives an overview of the features that we identified. In the following parts, we deeply analyzed these features and explained why we believe that they may be indicative features of DNS tunnel behavior.

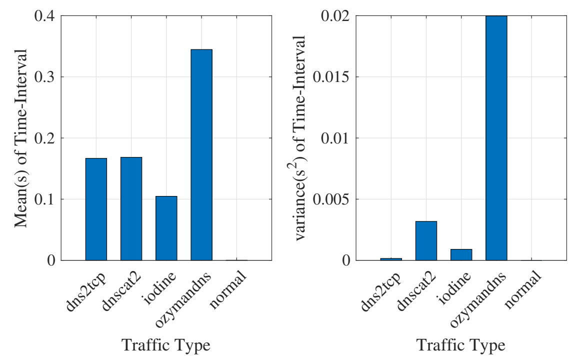


Fig. 4: Comparison of normal DNS and tunnel traffic on Time-Interval

To ensure our insights, we construct feature analysis dataset. We select 1000 request/response(req/res) pairs from an ISP RDNS server traffic as normal DNS traffic. Tunnel traffic produce by tunnel tools were extracted from Intranet RDNS server. We also select 1000 request/response pairs from tunnel traffic produced by four kinds of tunnel tools respectively to compare with normal DNS traffic. Every set of 1000 pairs has same source IP address, destination IP address and intended query domain.

1) Time-Interval: This feature set includes mean and variance of time intervals. A time interval refers to the time interval between two successive DNS query packet send from a client. For reducing the load time on the Domain Name System servers, caching results locally or in RDNS servers is a standard practice in implementing domain name resolution in applications. When a DNS response is provided with a TTL value included, the receiving RDNS server will cache that record for the time (in seconds) specified by the TTL. Then if RDNS receives a request from a host before the expire time, RDNS simply reply with the already cached resource record rather than retrieve it from the Authoritative DNS server again. In contrast, when DNS tunnel tools work in relay mode, the query domain name can not be generated in a period of TTL time. Every request that RDNS receives is forwarded to Authoritative DNS server and the corresponding response is returned from Authoritative DNS server because the domain name can not be found in RDNS cache. Accordingly, the time-interval of tunnel traffic is longer than time-interval of normal DNS traffic When the domain name is retrieved many times.

To validate our claims above, we use feature analysis dataset to calculate the mean and variance of time-interval. As shown in Figure 4, Normal DNS traffic has a lower value than tunnel traffic in both mean and variance of time-interval.

2) Request Packet Size: To transfer data to the destination, DNS tunnel tools encode secret information into a subdomain string and construct queried domain name with that string and a high-level domain name controlled by Authoritative DNS server. Tunnel application makes a request for resolving

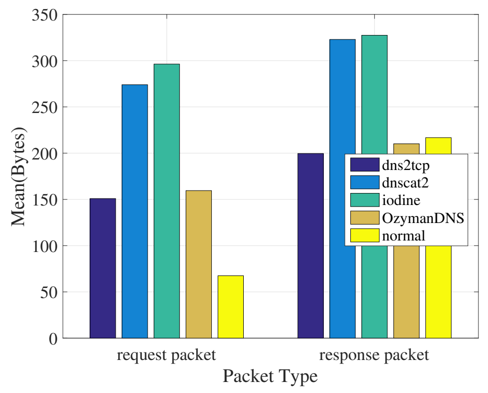


Fig. 5: Packet size mean comparison of normal DNS and tunnel traffic

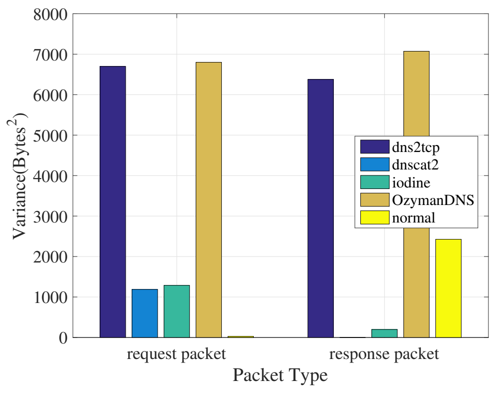


Fig. 6: Packe size variance Comparison of normal DNS and tunnel traffic

that domain name, Authoritative DNS server will receive the request and get the information by decoding the subdomain string. Longer domain name almost means more information. Attackers prefer making use of longer domain name to transfer more data as far as possible. This is why tunnel traffic has a larger request packet size than normal DNS traffic.

In the past approaches [8], researchers use both request packet size and response packet size as features. However, we found request packet size is a behavior feature to distinguish normal DNS traffic and tunnel traffic while response packet size is not. we think the reason is as follow. The answer section in a DNS message has multiple resource records of the queried domain name if it has multiple IP addresses associated. Figure 5 shows the mean of request packet size and response packet size respectively. The means of request packet size produce by four kinds of DNS tunnel tools are all above 150 bytes and the mean of request packet size from normal DNS traffic is between 50 and 100. Moreover, the results clearly show that the mean of response packet size is not a typical feature. As shown in Figure 6 Tunnel traffic has a far higher variance value of request packet size than normal DNS traffic except for traffic produced by DNScat2. DNScat2 produces equal size request packet and response packet over a certain period of time.

TABLE II: Example of subdomains extracted from normal DNS traffic and DNS tunnel traffic

Type	Subdomain
Normal Traffic	slide.mil.news
Tunnel Traffic	c2d2a57d30a40e3d4023fbb2e8f397023.c9156

3) Subdomain Entropy: Claude Shannon introduced the concept of Entropy into his research. Shannon entropy H is given by the formula:

$$H = -\sum_{i}^{n} p_i log_b(p_i) \tag{1}$$

Here, b is the base of the logarithm used and p_i is the probability of entropy unit. The higher the entropy, the more randomness. The normal domain names are selected carefully by human and they are recognizable and memorable to human. Therefore, the domain names follow the Zipf's law closely. Table II shows the example of subdomain extracted from two types of traffic. The subdomains generated by DNS tunnel tools are encoded by various encoding. The encoded subdomains are random and meaningless strings.

We extracted queried subdomains from one-hour-long DNS traffic collected from an ISP DNS server. The domains of extracted subdomains are included in the list of Alexa Top 1000 Global sites [27]. Subdomains produced by DNS tunnel tools collected from our Intranet DNS server. The subdomain unigram character frequency distribution is shown in Figure 7. It shows that subdomains produced by tunnel tools are more random. In this paper, we also use subdomain bigram and trigram to calculate entropy.

4) Record Type: Resource record has an important field of type [28]. Record type provides users with a particular type of information associated with queried domain name. There are many types of standard resource record. The common record types include A, NS, PTR, CNAME, MX and so on. There are also other less common record types containing TXT, NULL and others. Different record type gives a different hint of intended use. For example, A resource record maps a domain name to an IPv4 address and MX resource record is used to specifies a mail exchange server for a domain. After analysis of an-hour-long DNS dataset, we observed that A records constituted 70-80% of all traffic and AAAA records accounted for 8-10%. Furthermore, PTR records occupied about 6% and TXT, CNAME, NS accounted for 1%, respectively. The result of the analysis shows that common record types occupy most part of normal DNS traffic. In contrary, DNS tunnel tools prefer using uncommon record type to encapsulate information because several record types are expected to provide larger bandwidth. In the existing implements of DNS tunnel tools, several DNS record types are supported. For example, the author of iodine suggests that users give NULL type and PRIVATE high priority.

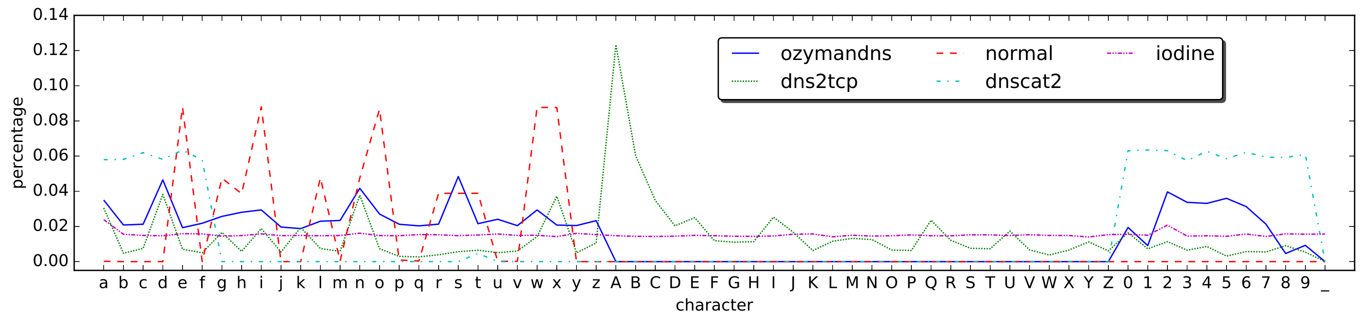


Fig. 7: Subdomains unigram character frequency distribution

TABLE III: Detection Performance using different algorithms about windows size N [%]

N	Decision Tree			Support Vector Machine			Logistic Regression		
	Accuracy(%)	Precision(%)	Recall(%)	Accuracy(%)	Precision(%)	Recall(%)	Accuracy(%)	Precision(%)	Recall(%)
1	97.92	100.00	95.89	99.66	100.00	99.32	99.47	100.00	98.95
5	97.94	100.00	95.92	99.66	100.00	99.32	99.36	100.00	98.73
10	98.04	100.00	96.12	99.28	100.00	98.58	99.65	100.00	99.30
15	98.04	100.00	96.13	99.77	100.00	99.54	99.69	99.92	99.48
20	99.77	99.89	99.65	99.66	100.00	99.33	99.81	99.98	99.64

VI. Evaluation

After analyzing deeply the features and presenting our approach in details, we now proceed to conduct our experiment and evaluate the performance. In this section, we first introduced our dataset and performance metrics. Then we evaluated the detection performance using the features we described in Section V. Finally, we compared the performance of the previous approach.

A. Data Collection and Performance Metrics

Our dataset includes two parts: training set and testing set. Every set of traffic includes two categories of traffic: real normal DNS traffic collected from an ISP DNS server and tunnel traffic collected from Intranet DNS server. The domains of normal DNS traffic is included in the list of Alexa Top 1000 Global sites [27]. The tunnel traffic was produced by DNS tunnel tools including dnscat2, iodine, dns2tcp and OzymanDNS. In the dataset, the training set contains 1.5 million req/res pairs of normal DNS traffic and 1.5 million req/res pairs of tunnel traffic. The testing set contains 1.15 million req/res pairs of normal DNS traffic and 1.15 million req/res pairs of tunnel traffic.

In the experiment, we used three metrics: Accuracy, Recall and Precision. For the binary-classification, we can classify the samples into four categories according to their real category and classifier prediction category: True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN). And the formulas of Accuracy, Recall and Precision are as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

$$Precision = \frac{TP}{FP + TP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

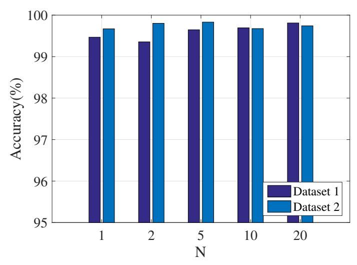
B. Parameters

When evaluating the performance of Detecting tunnel traffic, we set the following parameters to control:

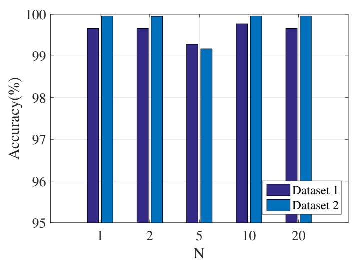
- **Window Size N:** In the following experiment, we generated samples using sliding window (not TCP sliding window) method which moves from the first request/response pair of the traffic to the final pair. Each time the sliding window moves one step forward. Sliding window method requires a setting for windows size N. The req/res pairs in the window have same source IP address, destination IP address and intended query domain. We generated every sample using N req/res pairs.For one purpose of tweaking hyperparameter N, we designed experiments with various N, including 1, 2, 5, 10, 20.
- **Binary-Classification Model:** To compare the performance of different Binary-Classification model in our approach, we also conducted experiments using three kinds of algorithms including Logistical Regression, Support Machine Vector and Decision Tree.

C. Detecting Performance

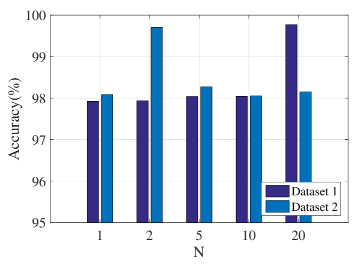
In our first experiment, We used the dataset to evaluate the performance of detecting tunnel traffic. We trained our binary-classifier using training set, then evaluated the classification results using testing set. In the training stage and testing stage, We use all of the features introduced in Section V. From Table III, we can see that the performance on SVM is better than other two algorithms. After analyzing 1 req/res pair, our approach based on SVM reaches the high detection accuracy rate of 99.66% with 99.32% recall rate. Moreover, we can see that the performance is better when the window size is larger than 1 whatever the algorithm. However, the detection performance may be lower when N is larger. For example,



(a) Logistic Regression

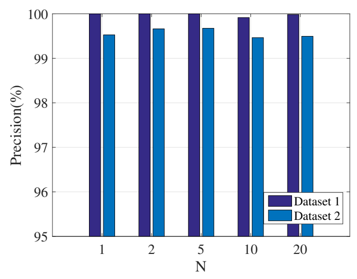


(b) Support Vector Machine

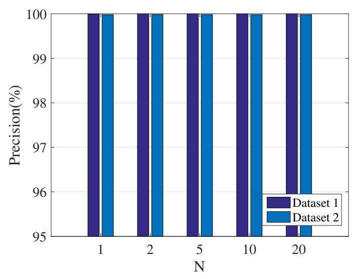


(c) Decision Tree

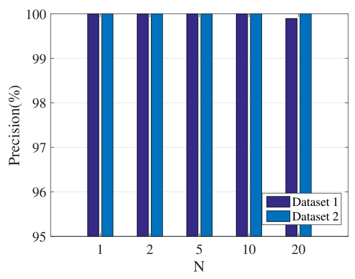
Fig. 8: Accuracy comparison of dataset 1 and dataset 2



(a) Logistic Regression

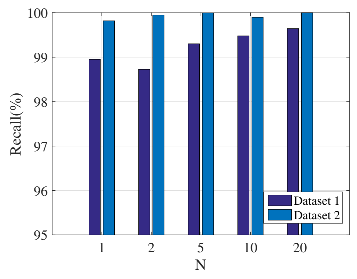


(b) Support Vector Machine

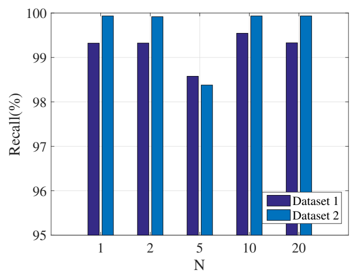


(c) Decision Tree

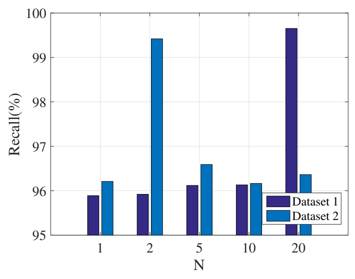
Fig. 9: Precision comparison of dataset 1 and dataset 2



(a) Logistic Regression



(b) Support Vector Machine



(c) Decision Tree

Fig. 10: Recall comparison of dataset 1 and dataset 2

when N is equal to 10, the performance using SVM algorithm is worse than the performance when N is equal to 5.

From Table III, we also can see that the precision is very high and recall is lower. High precision indicates the normal traffic is classified very correctly, whereas Low recall indicates that small part of tunnel traffic is not classified correctly. After studying the tunnel traffics, we found the tunnel tools generated a kind of traffic called tunnel-controlling in the process of a tunnel. The tunnel-controlling traffic has two purposes. One is to prevent the server timeouts in a period without tunneled data traffic and another purpose is to ensure the DNS tunnel server have a DNS request handy when there are data has to be sent from the server to client. So the DNS tunnel client has to send a request every several second's interval. The tunnel-controlling traffic has some similar features to normal DNS traffic. For example, the request packet size may be small and subdomain may be short. This is why some tunnel traffic is classified into normal traffic.

There is a small part of tunnel-controlling traffic in the tunnel training traffic set of dataset 1 which is generated in the process of transferring file. To solve the problem of low recall rate, we designed the dataset 2 and we just replaced half of the tunnel traffic with extracted tunnel-controlling traffic in the training set. The performance comparison between dataset 1 and dataset 2 is shown in Figure 8, Figure 9 and Figure 10.

Figure 8 shows the accuracy comparison. We can see that the performance is apparently better after we added the tunnel-controlling traffic into the training set in most windows size. After analyzing 1 req/res pair, our approach using SVM achieved high accuracy of 99.96%, 99.98% precision, and 99.93% recall. Figure 9 shows that the precision could be worse in Logistic Regression based on the dataset 2. Figure 10 shows the recall comparison between dataset 1 and dataset 2. We can see that the recall is clearly higher after we added the tunnel-controlling traffic into the training set.

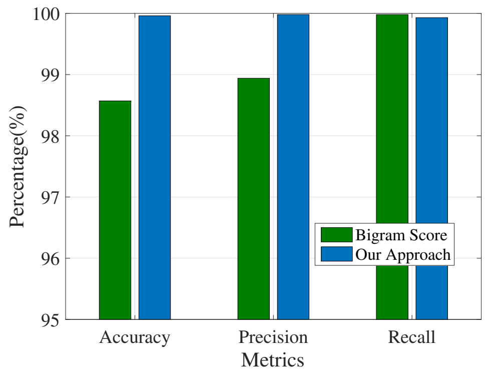


Fig. 11: Result Comparision

D. *Results Comparison*

In this experiments, we compare the detection results of our approaches against an approach based on Bigram [18]. It proposes a scoring mechanism based on Bigram. It scores domain names and set a threshold to distinguish DNS tunnel and normal traffic. We use the testing set to evaluate the approach, the comparison of performance with our approach is shown in Figure 11.

Among the three metrics, we can see that our approach performs better than the approach based on Bigram. In particular, our approach reaches higher accuracy and higher recall.

VII. Conclusion

In this paper, we proposed a binary-classification based approach to detect the DNS tunnel traffic. After deeply analyzing the behavior of DNS traffic, we extracted 4 kinds of features (totally 18 features) to build detection classifier. Our system sniffs the DNS traffic of RDNSs for detection. In the off-line training stage, the classifier is trained using labeled traffic. Then the classifier is used to determine whether the traffic from sniffer contains tunnel traffic. The experiments with real DNS traffic show that our method can identify the DNS tunnel traffic accurately and the precision is 99.98% and Recall rate is 99.93%.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61303261 and Grant 61402472.

References

[1] P. V. Mockapetris, “Domain names: Implementation specification,” 1983.

[2] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, and N. Pohlmann, “On botnets that use dns for command and control,” in *Proceedings of European Conference on Computer Network Defense*, 2011, pp. 9–16.

[3] “iodine,” https://github.com/yarrick/iodine, 2016.

[4] HSC, “dns2tcp,” http://www.hsc.fr/ressources/outils/dns2tcp/index.html.en, 2010.

[5] “dnscat2,” https://github.com/iagox86/dnscat2, 2016.

[6] D. Raman, B. De Sutter, B. Coppens, S. Volckaert, K. De Bosschere, P. Danhieux, and E. Van Buggenhout, “Dns tunneling for network penetration,” in *International Conference on Information Security and Cryptology*. Springer, 2012, pp. 65–77.

[7] M. S. Sheridan and A. Keane, “Detection of dns based covert channels,” in *ECCWS2015-Proceedings of the 14th European Conference on Cyber Warfare and Security 2015: ECCWS 2015*. Academic Conferences Limited, 2015, p. 267.

[8] M. Aiello, M. Mongelli, and G. Papaleo, “Supervised learning approaches with majority voting for dns tunneling detection,” in *International Joint Conference SOCO14-CISIS14-ICEUTE14*. Springer, 2014, pp. 463–472.

[9] S. Zander, G. Armitage, and P. Branch, “A survey of covert channels and countermeasures in computer network protocols,” *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.

[10] D. D. Wenkee Lee, “Next-generation dns monitoring tools,” *Cyber Security Division 2012 Principal Investigators Meeting*, 2012.

[11] David, “A traffic-analysis approach to detecting dns tunnels,” http://blog.vorant.com/2006/05/traffic-analysis-approach-to-detecting.html, 2006.

[12] S. Jaworski, “Using splunk to detect dns tunneling,” *SANS Institute InfoSec Reading Room*, 2016.

[13] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, and D. Dagon, “From throw-away traffic to bots: Detecting the rise of dga-based malware.” in *USENIX security symposium*, vol. 12, 2012.

[14] C. E. Shannon, “Prediction and entropy of printed english,” *Bell Labs Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[15] D. A. L. Romaña and Y. Musashi, “Entropy based analysis of dns query traffic in the campus network,” in *Proceedings for The 4th International Conference on Cybernetics and Information Technologies, System and Applications (CITSA 2007), Orlando, FL, USA*, 2007, pp. 162–164.

[16] P. Butler, K. Xu, and D. D. Yao, “Quantitatively analyzing stealthy communication channels,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2011, pp. 238–254.

[17] K. Born and D. Gustafson, “Detecting dns tunnels using character frequency analysis,” *arXiv preprint arXiv:1004.4358*, 2010.

[18] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, “A bigram based real time dns tunnel detection approach,” *Procedia Computer Science*, vol. 17, pp. 852–860, 2013.

[19] M. Himbeault, “A novel approach to detecting covert dns tunnels using throughput estimation,” Ph.D. dissertation, University of Manitoba, 2014.

[20] J. J. Guy, “dns part ii: visualization,” http://armatum.com/blog/2009/dns-part-ii/, 2009.

[21] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, “Flow-based detection of dns tunnels,” in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2013, pp. 124–135.

[22] T. van Leijenhorst, K.-W. Chin, and D. Lowe, “On the viability and performance of dns tunneling,” 2008.

[23] G. Farnham and A. Atlasis, “Detecting dns tunneling,” *SANS Institute InfoSec Reading Room*, pp. 1–32, 2013.

[24] K. Born, “Psudp: A passive approach to network-wide covert communication,” *Black Hat USA*, 2010.

[25] “tcp-over-dns,” http://analogbit.com/software/tcp-over-dns/, 2008.

[26] “Ozymandns,” https://dankaminsky.com/2004/07/29/51/.

[27] “Alexa web information company,” http://www.alexa.com/topsites, 2016.

[28] P. Mockapetris, “Rfc 1035domain namesimplementation and specification, november 1987,” *URL http://www. ietf. org/rfc/rfc1035. txt*, 2004.