Research on Application of Feature Analysis Method in DNS Tunnel Detection

Research on Application of Feature Analysis Method in DNS Tunnel Detection

Yong Liu[1, 3] and Xiangnan Gou[2]

[1]School of Information Engineering, Lingnan Normal University, Zhanjiang, China.
[2]Software School, Xiamen University, Xiamen Fujian, 361005, P.R. China.
[3]Department of Computer Engineering, Changji University, Changji, China.
Email: liuyong@lingnan.edu.cn

Abstract. Traditional DNS tunnel detection methods based on load analysis and traffic monitoring have a high false positive rate and cannot effectively respond to new DNS tunnel attacks. To this end, a log-based statistical method is proposed to detect DNS tunnel attacks. Compare and analyse the differences between DNS tunnel attack behaviours and normal DNS parsing behaviour from the perspective of DNS sessions, extract the multi-dimensional features dominated by cache hit ratios, compose DNS session evaluation vectors, and use random forest classification algorithms to construct DNS session evaluation vector detection classifiers A DNS tunnel attack detection model based on characteristic statistical behaviours is established. The actual test results show that this method has a small false positive rate and low false negative rate, and it also has a high detection ability for unknown DNS tunnel attacks.

1. Introduction

DNS service is the infrastructure of the Internet. By default, firewalls and intrusion detection systems will not detect and intercept DNS data. Therefore, DNS tunnels are widely used by attackers to steal information [1]. DNS tunnel refers to the use of the DNS protocol to establish a tunnel through DNS queries to achieve data transmission. Common DNS tunnel software includes iodine, dnscat2, dns2tcp [2][3] and so on. The DNS tunnel was originally used to obtain free WIFI [4]. In recent years, there have been more other incidents that use DNS tunnels to endanger network security. In the XshellGhost incident in August 2017, a number of NetSarang-owned software were implanted with advanced backdoors. DNS tunnelling is one of its communication methods, which directly led to users using its software becoming victims of remote control. Therefore, the research on DNS tunnel detection method is of practical value. The existing research methods are mainly divided into the following two ways according to the adopted characteristics.

The first is load-based analysis: the characteristics used in this method include request response size, domain name characteristics analysis (such as frequency of characters contained, domain name length, information entropy, ngram, etc.), record type, etc. [5][6][7]. The disadvantage of this method is that it will generate false positives for URL-encoded domain names, etc., and the software of the DNS tunnel can control the generated domain name length, character frequency, and other characteristics.

The second is based on the characteristics of network traffic: this method mainly uses DNS traffic characteristics. For example, the mechanism of analysing network flow data to detect DNS anomalies [8], based on basic characteristics such as packet size and packet arrival time interval [9], Luo et al. [10] identify DNS sessions through five-tuples. Feature extraction to detect DNS tunnels. Under the

currently researched detection methods, there are still false positives, and existing DNS tunnel construction software such as Dnscat can control the sending rate of query requests.

By analysing and summarizing the DNS tunnel detection methods in the existing literature, the existing research methods still have a high false alarm rate while ensuring accuracy. Therefore, this paper is based on the previous research methods, using DNS server query logs as the data source, mining the statistical characteristics mainly based on the cache hit rate, and constructing a DNS tunnel detection method with a lower false positive rate.

2. Feature Analysis

Although some detection methods in the existing literature have achieved a high accuracy rate, there will still be some false positives. For example, the methods in references [10][11] have false positives for normal domain names. Therefore, based on the shortcomings in the current method, this paper extracts new features such as cache hit rate and constructs a new detection method. This experiment first performs feature extraction based on the collected logs, expresses each log as a feature vector, and uses a random forest algorithm to build a DNS tunnel detection model.

2.1. Feature Construction

This paper analyses the difference between normal traffic and logs generated by tunnel traffic, and the features to be used in the proposed DNS tunnel detection method are shown in Table 1. The features in this paper are based on the extraction of secondary domains. The secondary domain is defined as a domain name with only two layers or two domain name tags [12]. Suppose there is a second-level domain D. There are m domain names under this second-level domain. The list of domain names is $d_D = \{d_1, d_2, \dots, d_m\}$. There are n logs of the query domain name in this set, and the log list is $logs_D = \{l_1, l_2, \dots, l_n\}$.

Table 1. Feature list.

Symbol	Implication
$\boldsymbol{E}$	Second-level domain entropy
$\boldsymbol{P_t}$	A/AAAA resource type query ratio
$\boldsymbol{C_s}$	Count of subdomains under second-level domains
$\boldsymbol{P_s}$	Unique query ratio in the second-level domain
$\boldsymbol{L}$	Domain name length
$\boldsymbol{L_{mvd}}$	Maximum vowel distance
$\boldsymbol{P_c}$	Cache hit ratio

2.1.1. Entropy of second-level domain names

Let the frequency of each character in the domain name d_i to be calculated be $P_d = \{p_1, p_2, \dots, p_k\}$. From the information theory, the entropy of this domain name is calculated as follows:

$$H(d_i) = -\sum_{i=1}^{k} p_i \times log(p_i) \quad (1)$$

After the entropy of the domain name is obtained, the average value of the entropy of the domain name in the same secondary domain is used as the entropy value of the secondary domain, then:

$$E(D) = \frac{\sum_{i=1}^{m} H(d_i)}{m} \quad (2)$$

Where m is the number of subdomains under the second-level domain D. Entropy is used to describe the degree of disorder of the object. The DNS tunnel is generated after the information is encrypted, so the entropy of the domain name is much larger than that of the normal domain name.

2.1.2. A/AAAA resource type query ratio

It can be known from the assumption that there are n logs in a secondary domain D, which are $logs_D = \{l_1, l_2, \dots, l_n\}$. For log l_n, then:

$$atype(l_i) = \begin{cases} 1 & \text{if log is A/AAAA} \\ 0 & \text{if log is not A/AAAA} \end{cases} \quad (3)$$

$$P_t(D) = \frac{\sum_{i=1}^{n} atype(l_i)}{n} \quad (4)$$

According to the reference [13], 99.4% of the query record types of DNS queries are A, AAAA, and PTR records. However, DNS tunnel communication often uses TXT and SRV records for communication, so DNS tunnels are used for communication the P_t value under the domain name is much higher than the normal domain name.

2.1.3. Counting Subdomains in Second-Level Domains

It can be known from the assumption that there are m domain names under a certain secondary domain D, and the domain name set is $d_D = \{d_1, d_2, \dots, d_m\}$, then:

$$C_s(D) = count\big(unique(d_D)\big) \quad (5)$$

Among them, $unique(d_D)$ represents the de-duplicated domain name set, and $count\big(unique(d_D)\big)$ represents the number of de-duplicated domain name sets.

2.1.4. Unique query ratio in the second-level domain

This feature evolved from C_s. There are n logs in a secondary domain D, which are $logs_D = \{l_1, l_2, \dots, l_n\}$. Calculated as follows:

$$P_s(D) = \frac{C_s(D)}{n} \quad (6)$$

If the second-level domain is the domain that the DNS tunnel uses to communicate, it is determined to send a large number of unique query requests, so the P_s ratio of the tunnel domain is larger than the normal second-level domain.

2.1.5. Domain name length

Suppose there is a second-level domain *D*, and there are m domain names under *D*, and the set of domain names is $d_D = \{d_1, d_2, \dots, d_m\}$, then the length of the second-level domain name is:

$$L(D) = \frac{\sum_{i=1}^{m} len(d_i)}{m} \quad (7)$$

The $len(d_i)$ is the length of the domain name d_i minus the length of the secondary domain *D*.

2.1.6. Maximum vowel distance

If the domain name to be detected is d_i, and the vowel character set is $\{'a','e','i','o','u'\}$, the distance of each vowel character is counted in d_i. Define the vowel distance as l_v. For d_i after removing the second-level domain, if the character has no vowels, the vowel distance is the distance from the character to the end of the string; otherwise, the vowel distance of the character is the distance between two vowel characters. Therefore, the vowel distance set in the domain name d_i to be detected is $L_v(d_i) = \{l_{v1}, l_{v2}, \dots, l_{vk}\}$, and the vowel distance of the secondary domain D to which it belongs is:

$$L_{mvd}(D) = \frac{\sum_{i=1}^{m} max(L_v(d_i))}{m} \quad (8)$$

For example, if the domain name $d = 'selected.\square icloud.com'$ has a vowel distance set of $L_v(d) = \{1,2,1\}$, the vowel distance of this domain name is $max(L_v(d)) = 2$. Reference [14] used vowel pitch to detect changing prefix domain name attacks. Previously, there were methods to judge the readability of domain names using the proportion of vowel characters in domain names [15]. Readability is whether the domain name is easy to pronounce. The pronunciation characteristics of English words are related to syllables. Tunnel domain names are generally generated through encryption, which is highly random and does not pay attention to pronunciation. Normal domain names usually guarantee the rhythm of pronunciation. Therefore, the vowel pitch of normal domain

names is smaller than that of tunnel domain names. Figure 1 is the L_{mvd} probability distribution of the normal domain name in the experimental data set, and Table 2 is the average L_{mvd} of the three DNS tunnel domain names used in the experiment.

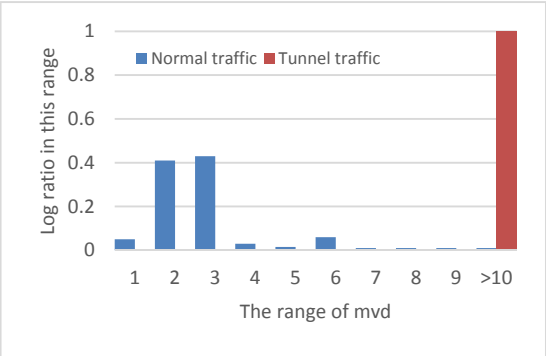

Figure 1. The L_{mvd} distribution of normal traffic and tunnel traffic.

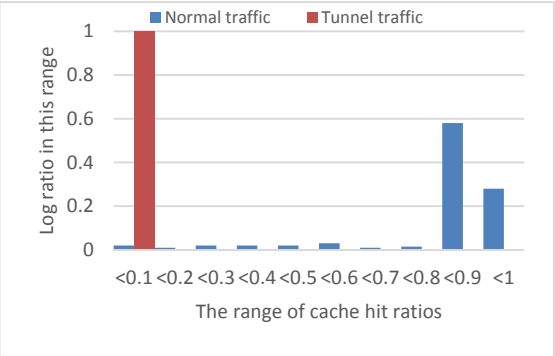

Figure 2. The P_c distribution of normal and tunnel data.

2.1.7. Cache hit ratio

Assume that within time t, the total number of queries for a certain secondary domain *D* is n times, and the log list is $Log_D = \{l_1, l_2, \dots, l_n\}$. The number of hits to the cache is defined as follows. Due to the DNS cache mechanism, each time a query is initiated, the query is first taken from the cache in the DNS server. If the query is the result obtained from the cache, it is considered to hit the cache once. Assuming that a secondary domain *D* hits the cache n_c times in total, the cache hit rate of the secondary domain *D* is:

$$P_c(D) = \frac{n_c}{n} \tag{9}$$

The DNS service has a caching mechanism. There is a TTL value in the data returned during the DNS query, which represents the time that the query record can exist in the cache. A normal domain name will set a higher TTL value to improve DNS query speed, and the query domain name will not change in general, and the cache hit rate is high. For DNS tunnels, on the one hand, its communication method determines it's The domain names used for communication are different, so caches are rarely hit, and the packet capture analysis in the experiment shows that the TTL in the DNS tunnel packets is very small, so even if the DNS server sets a mandatory cache, the cache under the tunnel domain name hit The rate is also extremely low. Based on the data used in this experiment, the distribution of cache hit rates for normal domain names and tunnel domain names is shown in Figure 2.

2.2. Detection Model

According to the seven characteristics of the DNS tunnel analysed in the previous chapter, each DNS query log is represented as the following quadruple <*Q, T, C*>, where *Q* is the query domain name, *T* is the query type, and *C* is the client who issued the query IP. After feature extraction, the feature vector corresponding to each secondary domain is as follows: $< E, P_t, C_s, P_s, L, L_{mvd}, P_c >$.

After investigating the previous methods, we decided to use the random forest algorithm as the classification model for this detection method [5]. The detection algorithm is a binary classification problem, which divides the data into tunnel data and normal data. Random forest uses multiple decision trees to train and predict the data, puts the input data on each decision tree, classifies the data and scores each feature, and finally integrates all the classification voting results. The category is specified as the final output.

2.2.1. Information gain algorithm

The random forest algorithm uses the idea of ensemble learning, which is a classifier composed of multiple CART (classification and regression tree). The decision tree determines the order in which

each feature is selected by calculating the information gain of each feature. Suppose that the sample to be classified has M category labels: $K = \{K_1, K_2, \dots, K_M\}$, and each sample has N features: $T = \{T_1, T_2, \dots, T_N\}$. According to information theory, information entropy indicates the degree of disorder of the data, and the information entropy of category K is defined as:

$$H(K) = -\sum_{i=1}^{M} p(K_i) \times log\big(p(K_i)\big) \tag{10}$$

For a certain feature, the added feature T_i is discrete. It is assumed that the feature T_i has z values, that is $T_i = \{T_{i1}, T_{i2}, \dots, T_{iz}\}$. The information gain is obtained from the information entropy, which represents the information of the known feature T_i, which makes the class K. The degree of reduction of data uncertainty, then the information gain is defined as:

$$IG(K|T_i) = H(K) - H(K|T_i) \tag{11}$$

Represents the conditional entropy of a category K in a given condition:

$$H(K|T_i) = -\sum_{x}^{M} \sum_{j}^{z} p\big(K_x, T_{ij}\big)\, log\, p\,\big(K_x, T_{ij}\big) \tag{12}$$

If the eigenvalues are not discrete and continuous, we can enumerate each of the two classifications and find the one with the greatest gain, that is:

$$IG(K|T_i) = \text{argmax}\big(IG(K|T\theta)\big) \tag{13}$$

2.2.2. Detection model training

Suppose the size of the original training set is N and the feature dimension of each sample is M. For each decision tree: Use the Bootstrap sample method to sample, randomly select N times to form a new training set; Choose a constant $m < M$. In each sampling process, only m features are randomly selected to form a new feature vector. When each node is split and grown, use the information gain algorithm described in section 2.1.1 to select one Features; Use the remaining $m - 1$ features on the newly-born tree node to continue to use the information gain algorithm to split and grow until the features are taken, reach the leaf node, and stop the process.

Assume that the current random forest is composed of the above k decision trees, and the current input sample is $S = (s_1, s_2, \dots, s_N)$. Each decision tree will output a corresponding result. The output set of the decision tree is $A = (a_1, a_2, \dots, a_k)$. There are two values for a_i:

$$a_i = \begin{cases} 0 & Input\ is\ tunnel\ traffic \\ 1 & Input\ is\ normal\ traffic \end{cases} \tag{14}$$

Therefore, for each sample s_i, the voting results of k decision trees:

$$V(s_i) = \sum_{j=1}^{k} a_j \tag{15}$$

The output of the random forest takes the mode of the output set of the decision tree. $V(s_i)/k > 0.5$ indicates that the current input is judged as tunnel traffic, and $V(s_i)/k \leq 0.5$ indicates that the current input is judged as normal traffic. The output is:

$$R(s_i) = \begin{cases} 1, \frac{V(s_i)}{k} > 0.5 \\ 0, \frac{V(s_i)}{k} \leq 0.5 \end{cases} \tag{16}$$

The DNS tunnel detection algorithm is shown in Algorithm 1.

Algorithm 1: Detection algorithm

Input: Sample set $S = (s_1, s_2, \dots, s_N)$, each sample is k-dimensional ($k = 7$).
Output: Output detection model and model accuracy.

```
trainSet, testSet = train_test_split(S)
rf = RandomForestClassifier(default)
rf.fit(trainSet)
score = GetRocAucScore(rf)
LOOP p in params_need_to_adjust
    rp = p.range
    g = GridSearchCV(rp)
    params.add(g.p)
END LOOP
rf = RandomForestClassifier(default)
score = GetRocAucScore(rf)
```

3. Experiments Analysis

3.1. Data Set

The data set used in the experiment is shown in Table 2:

Table 2. Experimental data set.

Sample source	Number
DNS server of a company	12971
Iodine	48
Dns2tcp	16
Dnscat2	32

First, collect access logs from a company's DNS server for eight time periods in a day, and manually check them as normal access logs. Take a client to access the sample with the most frequent access to the same second-level domain name for experiments. Take three types of DNS tunnel software idle and busy logs, intercept any four time periods, and collect and use the query records that it can support for data collection. A total of 24 DNS tunnel logs are trained for a total of 96. Tunnel samples. In this experiment, two types of DNS server cache hit rate collection were implemented, CoreDns and BIND. In order to obtain the DNS cache, hit rate, the source code of the two software was modified and the *hitcache* field was added to its output log:

$$hitcache = \begin{cases} 0 & Query\ misses\ cache \\ 1 & Query\ hits\ cache \end{cases} \tag{17}$$

3.2. Experimental Results

In To test the effectiveness of the algorithm in the Hadoop cluster, we performed a Speed up test. Speed up performs experiments for clusters with different number of nodes and tests the efficiency of the algorithm when increasing the number of nodes in the cluster. In the experiment, we fixed the amount of data to 1GB and used.

After the obtained data is extracted with features, it is vectorized into a seven-dimensional vector and saved as a .csv format. The data source is divided into a training set and a test set according to 7: 3, and the random forest model is trained. Initialize the various parameters in the random forest and select the default parameters. Then, perform a grid search on the random forest parameters, optimize the model, and finally perform a ten-fold cross-validation model. This article uses the receiver's operating characteristic curve area [16], that is, *roc_auc_score*. For evaluation, the scoring results of ten experiments are shown in Table 3. Using the average *roc_auc_score* of the experiment as the final

score of the model, the current model score is 99.1. After testing, all DNS tunnel traffic in the test set was detected, and the false positive rate was also 0%.

Table 3. Ten-fold cross-validation results.

Number of times	1	2	3	4	5	6	7	8	9	10
Score	99.1	99.0	99.2	99.6	99.1	98.9	99.0	99.0	99.1	99.0

3.3. Experimental Comparison

In order to evaluate the effectiveness of this model in detecting DNS tunnels, this article selects recently published literature for comparative experiments.

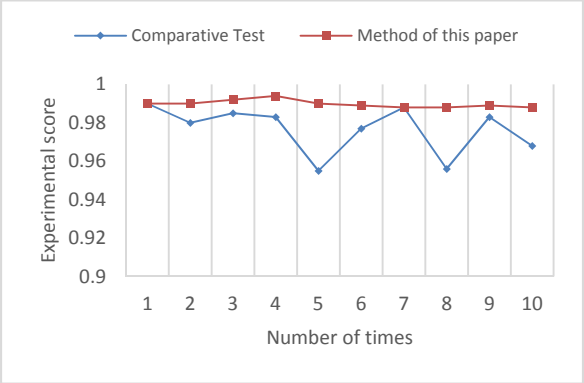

Figure 3. Comparison of experimental results.

We selected the method in reference [11] to conduct comparative experiments on DNS tunnel detection under the feature dimension. Using *roc_auc_score* to compare the two methods, the results are shown in Figure 3. From the comparison results, we can see that our proposed DNS tunnel detection model can reduce the false positives of normal domain names through the newly introduced features, that is, the accuracy rate is improved.

4. Conclusion

From the perspective of DNS server, this paper proposes a DNS tunnel detection method based on log statistical characteristics. However, existing DNS tunnel detection methods have found that there are still false positives. The detection method in this paper incorporates multi-dimensional features such as cache hit rate, and designs a DNS tunnel detection algorithm based on a random forest model. A comparison experiment with existing detection methods was carried out. The experiments prove that the method in this paper has higher accuracy and lower false positive rate in DNS tunnel detection.

5. References

[1] Born, Kenton, and David Gustafson. "Detecting dns tunnels using character frequency analysis." arXiv preprint arXiv:1004.4358 (2010).
[2] Lin, Huaqing, Gao Liu, and Zheng Yan. "Detection of application-layer tunnels with rules and machine learning." International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, Cham, 2019.
[3] Buczak, Anna L., et al. "Detection of tunnels in PCAP data by random forests." Proceedings of the 11th Annual Cyber and Information Security Research Conference. 2016.
[4] Kara, A. Mert, et al. "Detection of malicious payload distribution channels in DNS." 2014 IEEE International Conference on Communications (ICC). IEEE, 2014.
[5] Qi, Cheng, et al. "A bigram based real time DNS tunnel detection approach." Procedia Computer Science 17 (2013): 852-860.

[6] Yu, Bin, et al. "Behavior Analysis based DNS Tunneling Detection and Classification with Big Data Technologies." IoTBD. 2016.
[7] Liu, Jingkun, et al. "Detecting DNS tunnel through binary-classification based on behavior features." 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017.
[8] Mugali, Aditya Anand, Andrew W. Simpson, and Scott King Walker. "System and method for detecting DNS traffic anomalies." U.S. Patent No. 9,172,716. 27 Oct. 2015.
[9] Davis, Jonathan J., and Ernest Foo. "Automated feature engineering for HTTP tunnel detection." Computers & Security 59 (2016): 166-185.
[10] You-qiang, L. U. O., et al. "DNS tunnel Trojan detection method based on communication behavior analysis." Journal of ZheJiang University (Engineering Science) 51.9 (2017): 1780-1787.
[11] Nadler, Asaf, Avi Aminov, and Asaf Shabtai. "Detection of malicious and low throughput data exfiltration over the DNS protocol." Computers & Security 80 (2019): 36-53.
[12] Jianqiang, Yang , and J. Hongxi . "Using FQDN number of the second-level domain name to detect DNS-based covert channels." Computer Era (2016).
[13] Herrmann, Dominik, Christian Banse, and Hannes Federrath. "Behavior-based tracking: Exploiting characteristic patterns in DNS traffic." Computers & Security 39 (2013): 17-33.
[14] Kirchler, Matthias, et al. "Tracked without a trace: linking sessions of users by unsupervised learning of patterns in their DNS traffic." Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. 2016.
[15] Hao, Shuai, and Haining Wang. "Exploring domain name based features on the effectiveness of dns caching." ACM SIGCOMM Computer Communication Review 47.1 (2017): 36-42.
[16] WANG, Yun-Yun, and Song-Can CHEN. "A survey of evaluation and design for AUC based classifier." Pattern Recognition and Artificial Intelligence 1 (2011).