Detection of Application-Layer Tunnels with Rules and Machine Learning

Chapter · July 2019

DOI: 10.1007/978-3-030-24907-6_33

3 authors, including:

Huaqing Lin
Xidian University
4 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Zheng Yan
Xidian University
221 PUBLICATIONS **2,748** CITATIONS
SEE PROFILE

Detection of Application-Layer Tunnels with Rules and Machine Learning

Huaqing Lin[1], Gao Liu[1], and Zheng Yan[1, 2] (✉)

[1] State Lab of ISN, School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2] Department of Communications and Networking, Aalto University, Espoo 02150, Finland

answer3lin@qq.com; 986976804@qq.com; zyan@xidian.edu.cn

Abstract. Application-layer tunnels are often used to construct covert channels in order to transmit secret data, which is often applied to raise network threats in recent years. Detection of application-layer tunnels can assist identifying a variety of network threats, thus has high research significance. In this paper, we explore application-layer tunnel detection and propose a generic detection method by applying both rules and machine learning. Our detection method mainly consists of two parts: rule-based domain name filtering for Domain Generation Algorithm (DGA) based on a trigram model and a machine learning model based on our proposed generic feature extraction framework for tunnel detection. The rule-based DGA domain name filtering can eliminate some obvious tunnels in order to reduce the amount of data processed by machine learning-based detection, thereby, the detection efficiency can be improved. The generic feature extraction framework comprehensively integrates previous research results by combining multiple detection methods, supporting multiple layers and performing multiple feature extraction. We take the three most common application-layer tunnels, i.e., DNS tunnel, HTTP tunnel and HTTPS tunnel as examples to analyze and test our detection method. The experimental results show that the proposed method is generic and efficient, compared with other existing approaches.

Keywords: Application-layer Tunnels Detection, Machine Learning, DGA Domain Name, Feature Extraction.

1 Introduction

1.1 Background

Current networks are becoming more and more complex and easy to suffer from various network attacks. Many typical attacks, such as Trojan, Botnet, and Advanced Persistent Threat (APT) need to establish Command & Control (C&C) communication channels in order to hack a target network system. Therefore, it is possible to analyze and detect these threats during network communication phase for the purpose of blocking the communication process of network threats in real time and protecting the network system.

In order to avoid being identified, attackers usually build a covert channel in the form of an application-layer tunnel to implement C&C communications. Therefore, many kinds of network threats can be detected by identifying application-layer tunnels [1].

The application-layer tunnel refers to encapsulating data packets of a protocol into the payload of the data packets of an application layer protocol and is in a form of "Protocol over Protocol" [2-4]. We refer to the outer protocol data packet carrying the data as delivery packet and the protocol data packet carried in the inner layer of the delivery packet as payload packet. Due to the superiority of tunneling technology in secure transmission, protocol compatibility, firewall penetration, etc., the application-layer tunnel is often used to construct covert channels to transmit secret data. As shown in Fig. 1, a threat disguises as a normal application layer protocol to bypass security control policies of a firewall or an Intrusion Detection System (IDS) in order to carry arbitrary data in and out of an internal network. A malicious application-layer tunnel is a kind of network covert channels that assist attackers to communicate with intruded hosts in the Intranet [3, 4, 5], realize malicious remote access [3, 4, 6]; and steal traffic [5]. In recent years, more and more network threats used the application-layer tunnels to construct covert channels for covert communications, including C&C, desktop control, stealing private data and transmitting secret files.

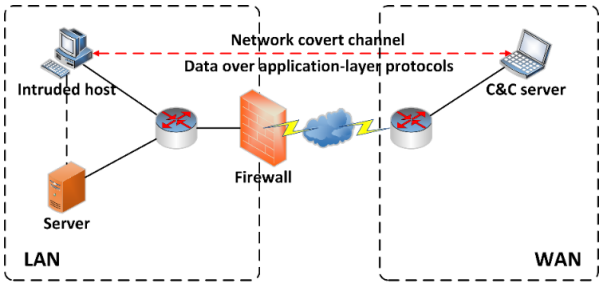

Fig. 1. The threat model of application-layer tunnels.

1.2 Three Methods of Tunnel Detection

The high communication traffic volume of application layer protocol and complicated business and services cause challenges to application layer tunnel detection. There are three detection methods in the current literature: feature signature-based detection [9], protocol anomaly-based detection [2, 6, 7, 9] and behavior statistics-based detection [2-6, 8-15]. The feature signature-based detection method is to detect tunnels by building a signature library and matching signatures in protocol data. This method suffers from a high false positive rate and low efficiency. In addition, this method cannot work on encrypted tunnels. The protocol anomaly-based detection method detects the tunnels by analyzing whether the communication data conform to the standard of normal network protocol. However, with the development of protocol camouflage technology, the identification rate of protocol anomaly-based detection becomes low. The behavior statistics-based detection method is widely studied, which detects tunnels by analyzing the behavior of network communication data. However, this method has some disadvantages caused by behavior analysis difficulty, complex modeling and poor real-time

performance. With the continuous enrichment of network services and the increasing complexity of network protocols, it is difficult for a single detection method to achieve high accuracy and low false positive rate. Therefore, we propose to integrate the advantages of these three kinds of detection methods to efficiently and comprehensively detect application-layer tunnels.

1.3 Rules & Machine Learning

Early intrusion detection methods employ anomaly detection or misuse detection for extracting rules to detect threats, intrusions and attacks. The rule-based detection method is simple to implement and is very useful for identifying obvious anomalies. However, it is difficult to describe complex attack modes. When the rules are complex, rule conflict issues might appear. In recent years, with the development of artificial intelligence, the application of machine learning and deep learning to network security has become a hot topic [16, 17]. Machine learning based detection can learn the features of normal and malicious samples. It can combine the advantages of detection and misuse detection to achieve high detection accuracy. The simple modeling of machine learning saves a lot of time and manpower. Moreover, it can describe high-dimensional features and mine potential association rules. However, its disadvantage is that modeling requires a large-scale labeled dataset, otherwise over-fitting often occurs. Usually, rules and machine learning were used separately, but we try to make use of the advantages of these two methods for detecting application-layer tunnels efficiently and accurately. Concretely, simple rules are used to filter obvious tunnels, thus improving the detection efficiency and real-time performance. The machine learning model can be applied to detect complex tunnels, thus ensuring high detection accuracy. Therefore, an efficient and accurate tunnels detection method can be proposed based on the above idea.

1.4 Contributions

In order to overcome the shortcomings of existing work, we propose a method to efficiently, comprehensively, and accurately detect application layer tunnels by applying both rules and machine learning. It consists of two parts, namely rule-based DGA domain name filtering and machine learning based on our proposed feature extraction framework for tunnel detection. When the domain name adopted by the communication data does not satisfy the DGA domain name filtering rule, these data are blocked directly. Otherwise, a machine learning model will be used to further analyze them. Specifically, the contributions of this paper can be summarized as follows:

- We propose an application-layer tunnel detection method by applying rules and machine learning. We employ a trigram model to design rule-based DGA domain name filtering, which can identify tunnels with obvious features for reducing the amount of data that need to be further processed during machine learning-based detection. Therefore, our method can greatly improve efficiency and real-time performance of tunnel detection. In terms of machine learning, we propose a generic feature extraction framework by combining multiple detection methods, supporting network layer, transport layer and application layer and performing multiple statistical and security-related features extraction for tunnel detection;

- We test the effectiveness of the proposed method by conducting experiments on common Domain Name System (DNS), Hyper Text Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) tunnels. Experimental results show that our proposed method is more generic and efficient compared with other existing works.

The rest of the paper is organized as follows. Section 2 gives a brief review of related work on application-layer tunnel detection. We provide detailed design of our detection method in Section 3. In Section 4, we evaluate and analyze the proposed method. Conclusions and future work are presented in the last section.

2 Related Work

Many related works only extract the statistical features of data packets behavior in network and transport layers [3, 4, 8, 10, 11-15]. The advantages of this kind of methods are easy to implement, high efficiency and can be applied into various tunnel detection. However, due to the lack of application layer features, their detection accuracy is low. On the contrary, the accuracy of feature extraction in application layer is high, but it also suffers from some other problems [2, 5-7, 9]. For example, Borders et al. [2] proposed an HTTP tunnel detection method named Web Tap to detect HTTP tunnels by analyzing application layer data content features and behavioral features. Its accuracy is high, but its real-time performance is poor and this method is not generic. Qi et al. [7] analyzed domain names in DNS packets based on a bigram model to detect DNS tunnels in real time. This method has low computational complexity and does not need flow recombination, so that it has high real-time performance. But when the character frequency of some normal domain names does not follow the Zipf's law, it easily produces false positive rate. Liu et al. [9] extracted four kinds of 18 features from the content and behavior of DNS tunnel traffic by considering network, transport and application layers in order to achieve a high detection accuracy. However, this method uses *n* query/response pairs to form a window as an analysis unit, which results in a large detection delay.

3 Proposed Detection Method

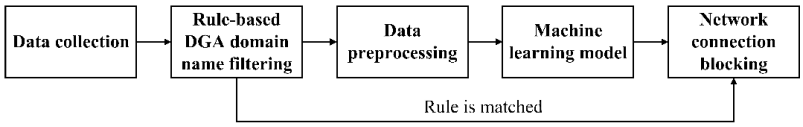

Fig. 2. Application-layer tunnel detection method.

We combine the feature signature-based detection method, the protocol anomaly-based detection method and the behavior statistics-based detection method and use machine learning and rules to analyze the collected data to identify the application-layer tunnels for preventing unauthorized data transmission. As shown in Fig. 2, the proposed application-layer tunnel detection method includes five modules: 1) data collection module.

This module is responsible for collecting communication data for subsequent analysis; 2) rule-based DGA domain name filtering module. This module first extracts the domain name from the communication data and then matches the domain name with the proposed filtering rule. If the matching is successful, a network connection blocking module will be notified to block the connection. Otherwise, the data will be transmitted to a data preprocessing module for further analysis; 3) data preprocessing module. This module is responsible for performing feature construction and feature extraction in order to process communication data to transfer them into the data format required by a machine learning model; 4) machine learning module. This module can classify the communication data to detect whether there is an application layer tunnel. When a tunnel is identified, the network connection blocking module is notified to block the network connection; 5) network connection blocking module. This module can cut off the network connection and block data transmission.

3.1 Data Collection

The first step in the proposed detection method is to collect data. After obtaining normal data and malicious data, we can further analyze the data and detect threats, intrusions, or attacks. In our study, normal network traffic data are collected through port mirroring in our university campus network. Because we mainly focus on DNS, HTTP and HTTPS tunnels as detection examples, we only need to collect their corresponding data. DNS, HTTP and HTTPS tunnel data are all collected in the university campus network and VPS (Virtual Private Server). Malicious tunnel communication data were simulated by applying some tunnel tools as listed in Table 1. As shown in Table 1, we collected 727052 DNS tunnel sessions by three DNS tunneling tools, 43442 HTTP tunnel sessions by five HTTP tunneling tools and 16129 HTTPS tunnel sessions by three HTTPS tunneling tools. The payload protocols over application-layer tunnels include remote access protocol, reverse remote access protocol, data transfer protocol, chat protocol and Remote Desktop Protocol (RDP).

Table 1. Application-layer tunnel data collection.

Delivery protocol	Tunnel tool	Payload protocol	Dataset size (sessions)
DNS	dns2tcp, dnscat2, iodine	Shell (SSH), Data (SSH), Chat (Ncat), Shell (Ncat), Reverse shell (Ncat), Data (Ncat), RDP	727,052
HTTP	ABPTTS, Httptunnel, reDuh, reGeorg, Tunna		43,442
HTTPS	Stunnel, Ghostunnel, Meterpreter		16,129

3.2 Rule-Based Detection

We design a rule to efficiently and accurately detect some application-layer tunnels with obvious features for reducing the amount of data needed for machine learning based detection. This filtering rule can be applied to most application-layer tunnels, thus having generality. DGA is a technical mean of generating C&C server domain names by creating random characters. It can bypass the detection of domain name blacklists by generating a large number of random domain names [18, 19]. DGA based

communication connection is flexible because attackers can change the domain name of C&C server through DGA to effectively avoid the detection based on domain name blacklist. This attack technology is applied to a variety of network threats, such as XShellGhost and CryptoLocker. DGA is also paired with application-layer tunnels to achieve higher concealment. DNS, HTTP and HTTPS tunnels often generate C&C server domain names through DGA to implement C&C communications. Therefore, we can identify DGA domain names to filter application-layer tunnels with obvious domain name characteristics.

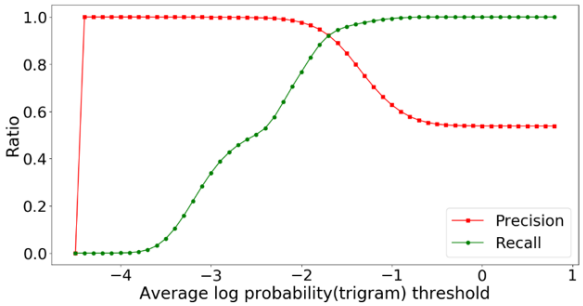

Fig. 3. Classification result of DGA domain names with trigram model.

In order to design an efficient DGA filtering rule, we do not consider deep learning models such as CNN and LSTM [19], but test the classification performance of DGA domain names by means of information entropy, bigram model and trigram model. The test dataset is comprised of Alexa global traffic top 1 million website [20] and 1164838 DGA domain names from 360 Netlab [21]. The experimental results show that the trigram model has the best classification performance for DGA domain names, as shown in Fig. 3. Generally speaking, if the accuracy of classification can reach 99.9% in a production environment, it can meet demands. Therefore, in order to meet the accuracy requirement, we choose -3 as the average logarithmic probability threshold of trigram model. In this case, the precision, recall and false positive rate are 0.99903, 0.33973 and 0.000383, respectively. That is to say, by adopting this rule, we can directly filter 33.973% of the application-layer tunnels, which greatly reduces the data preprocessing and prediction time of machine learning-based detection and saves computing resources. At the same time, the proposed rule can directly analyze and detect tunnels without reassembling packets and constructing analysis units (flows), thus improving the real-time performance of detection.

3.3 Machine Learning Model Construction

Generic Feature Extraction Framework. In order to construct a machine learning model that can detect multiple application-level tunnels, we first propose a generic feature extraction framework. Through the analysis of Section 1.2, we found that the existing three tunnel detection methods have their own problems. Moreover, many related

works mainly extract some statistical features from the network and transport layers and rarely extract security-related features from the application layer. In fact, these are the most discriminative features that affect the accuracy of tunnel detection. Therefore, we combine feature signature-based detection, protocol anomaly-based detection and behavior statistical-based detection to design a generic feature extraction framework that can extract security-related features and statistical features from the network, transport and application layers. As shown in Table 2, we introduce the proposed framework in detail according to different detection methods.

Table 2. A generic feature extraction framework for application-layer tunnel detection.

<table>
<tr><th colspan="3">Detection method</th><th>Protocol layer</th><th>Direction</th><th>Export value</th></tr>
<tr><td rowspan="6">Behavior statistics</td><td rowspan="2">Length</td><td>Absolute length</td><td rowspan="2">Network layer</td><td rowspan="3">Upstream</td><td>Max</td></tr>
<tr><td>Relative length</td><td>Min</td></tr>
<tr><td>Time</td><td>Relative time</td><td>Transport layer</td><td>Mean</td></tr>
<tr><td rowspan="2">Count</td><td>Absolute count</td><td rowspan="2">Application layer</td><td rowspan="2">Downstream</td><td>Variance</td></tr>
<tr><td>Relative count</td><td>Sum</td></tr>
<tr><td rowspan="2">Feature signature</td><td colspan="5">Key fields of request data header</td></tr>
<tr><td colspan="5">Key fields of response data header</td></tr>
<tr><td rowspan="3">Protocol anomaly</td><td colspan="5">Protocol request data anomaly</td></tr>
<tr><td colspan="5">Protocol response data anomaly</td></tr>
<tr><td colspan="5">Protocol interaction anomaly</td></tr>
</table>

Feature Signature based Detection Method. Different from traditional feature signature-based detection methods, we do not consider payload data here and do not need to build a signature library. The advantage of this modification is to solve the issue of payload data encryption and preserve user privacy, and to especially improve the detection efficiency. That is, in this method, we only record the key fields of application layer protocol header, including request and response (e.g., flag and TTL in DNS packets).

Protocol Anomaly based Detection Method. In this method, we take the request, response and interaction anomalies of the request and response as features. The request anomaly mainly refers to whether the fields of protocol request data have a match error. The response anomaly mainly refers to whether the fields of protocol response data have a match error. The protocol interaction anomaly refers to whether the response data satisfy the request made in the request data. For example, when the record of DNS response packet is A, the resource data should be IPv4. Otherwise, an anomaly is triggered.

Behavior Statistics based Detection Method. Behavior statistics-based detection method is studied widely and many related works only extract the features related to the behavior of data packets. Behavior statistics related features include packet size (length), arrival time and number (count). The size and number include absolute value

and relative value. However, the absolute value of time is meaningless for detection, and thus only the relative value of time is considered. Absolute value refers to the value of the data itself while the relative value refers to the correlation among these values. Additionally, the features are extracted from an application layer in addition to network and transport layers. Moreover, the data packets are upstream and downstream. Basically, each feature can derive five values: maximum, minimum, mean, variance and sum.

Application-Layer Tunnel Detection Model Construction.

DNS Tunnel Detection Model. First, we reassemble DNS packets into sessions. We define a DNS session as a query with a corresponding response. Based on the proposed generic feature extraction framework, we then extract 36 features for the DNS tunnel. Finally, a tree model based feature selection method is used to select 25 feature subsets to construct a DNS tunnel detection model.

HTTP Tunnel Detection Model. First, we reassemble HTTP packets into sessions. We define an HTTP session as a TCP stream that is divided by a 5-tuple (source IP, destination IP, source port, destination port and protocol type). Based on the proposed generic feature extraction framework, we then extract 79 features for the HTTP tunnel. Finally, a tree model based feature selection method is used to select 32 feature subsets for constructing the HTTP tunnel detection model.

HTTPS Tunnel Detection Model. First, we reassemble HTTPS packets into sessions. The definition of an HTTPS session is the same as the HTTP session that consists of a TCP stream. Based on the proposed generic feature extraction framework, we then extract 72 features for the HTTPS tunnel. Finally, a tree model based feature selection method is used to select 30 feature subsets for constructing the HTTPS tunnel detection model.

4 Experimental Results and Analysis

4.1 Machine Learning Model Evaluation

DNS Tunnel Detection Model Evaluation

Comparison of Different Classification Algorithms. First, we used six common machine learning algorithms to test DNS tunnel classification performance, including Gaussian Bayesian, Gaussian kernel SVM, C4.5 decision tree, random forest, GBDT and XGboost. As shown in Table 3, we can find that the classification results of the five machine learning algorithms except Gauss Bayes are very good.

Comparison of Our DNS Tunnel Detection with Related Work. Due to the different test datasets used in different papers, the experimental results in multiple papers are not comparable. Therefore, in order to further test the performance of our proposed method,

we implemented three existing detection methods and compared them with our proposed method based on the collected datasets. The comparison results of our DNS tunnel detection model based on random forest and behavior-based detection model in [9] are shown in Fig. 4. The behavior-based DNS tunnel detection model extracts 18 features. In the experiment, the window size is set to 5 and SVM with the best performance was selected as the modeling algorithm. As depicted in Fig. 4, we can see that our feature extraction framework based DNS tunnel detection model offers higher detection accuracy, consumes less time and has better real-time performance compared with the model in [9]. The window size of behavior-based DNS tunnel detection model in [9] is set to 5, which suggests its detection delay is 5 times of that of our proposed detection model because our proposed model only processes a DNS session but the model in [9] needs to process 5 sessions in a window. Based on the same dataset, our model needs to process five times as much data as the model in [9] does and the feature dimension of our model is higher. However, the processing performance of random forest is better than SVM, so that the computation time of our model is shorter than [9].

Table 3. Performance comparison of different classification algorithms for DNS tunnel detection.

Classification algorithms	Precision	Recall	Accuracy	F1 score	Training time/s	Testing time/s
Gaussian Bayes	0.73423	1.0	1.0	0.84513	1.6674	1.1981
SVM	0.99999	0.99836	0.99836	0.99917	194.5913	43.7776
C4.5	0.99999	0.9992	0.9992	0.9996	10.3534	0.5064
Random forest	1.0	0.99966	0.99966	0.99983	8.4319	0.9932
GBDT	0.99999	0.999	0.999	0.99949	487.3548	1.4265
XGboost	1.0	0.99981	0.99981	0.9999	50.3657	0.7284

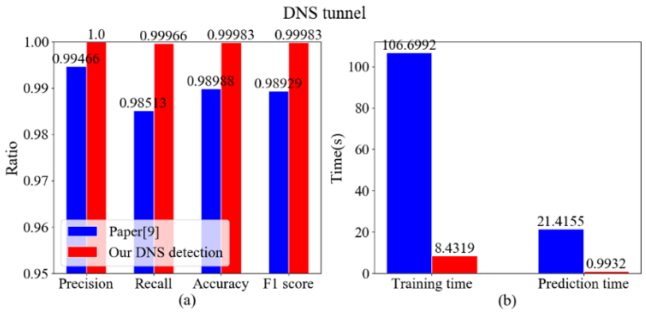

Fig. 4. Comparison of our DNS tunnel detection with the detection in [9].

HTTP Tunnel Detection Evaluation

Comparison of Different Classification Algorithms. As shown in Table 4, we can find that the classification results of the five machine learning algorithms except Gauss Bayes are very good.

Comparison of Our HTTP tunnel Detection with Related Work. In order to further test the performance of our proposed method, we implemented HTTP tunnel detection model based on statistical features in [10] and compared it with our proposed random forest based HTTP tunnel detection model based on the collected datasets. The comparison results of these two HTTP tunnel detection models are shown in Fig. 5. The statistical features based detection model selects 21 features and uses C4.5 decision tree as the modeling algorithm. As descripted in Fig. 5, we can see that our proposed HTTP tunnel detection model has higher detection accuracy compared with the model in [10]. This is because we extract security-related features from the application layer while the model in [10] only consider the features from the network and transport layers. However, the prediction time of our detection model is a bit longer, since C4.5 decision tree is faster than random forest.

Table 4. Performance comparison of different classification algorithms for HTTP tunnel detection.

Classification algorithms	Precision	Recall	Accuracy	F1 score	Training time/s	Testing time/s
Gaussian Bayes	0.5404	0.99414	0.5743	0.70018	0.7186	0.4912
SVM	0.97279	0.98712	0.97976	0.9799	11130.3015	396.2684
C4.5	0.99884	0.99894	0.99889	0.99889	8.733	0.1173
Random forest	0.99939	0.9985	0.99895	0.99895	3.9287	0.7579
GBDT	0.99615	0.99331	0.99474	0.99473	251.1598	0.5194
XGboost	0.9931	0.99471	0.99389	0.9939	18.7661	0.2602

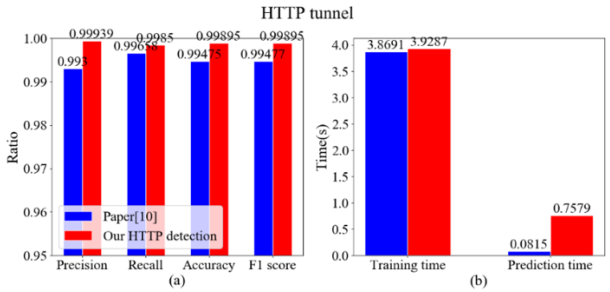

Fig. 5. Comparison of our HTTP tunnel detection with the detection in [10].

HTTPS Tunnel Detection Model Evaluation

Comparison of Different Classification Algorithms. As shown in Table 5, we can find that the classification results of the five machine learning algorithms except Gauss Bayes are very good.

Comparison of Our HTTPS tunnel Detection with Related Work. In order to further test the performance of our proposed method, we implemented HTTPS tunnel detection

model in [15] and compared them with our random forest based HTTPS tunnel detection model based on the collected datasets. The model in [15] selects 10 features and also use random forest as the modeling algorithm. As shown in Fig. 6, we can see that our HTTPS tunnel detection model has higher detection accuracy compared with the model in [15]. This is because we extract security-related features from the application layer while the model in [15] only considers the features from the network and transport layers. Since this method also uses random forest as the modeling algorithm, the training and prediction time of the two models is similar.

Table 5. Performance comparison of different classification algorithms for HTTPS tunnel detection.

Classification algorithms	Precision	Recall	Accuracy	F1 score	Training time/s	Testing time/s
Gaussian Bayes	0.89127	0.73891	0.77196	0.77078	0.0607	0.0741
SVM	0.99377	0.99729	0.99551	0.99552	11.2404	3.2969
C4.5	0.99879	0.99912	0.99895	0.99895	0.9605	0.0152
Random forest	0.99938	0.99949	0.99943	0.99943	0.8882	0.6313
GBDT	0.99831	0.99904	0.99867	0.99867	23.588	0.0687
XGboost	0.99873	0.99932	0.99902	0.99903	3.1652	0.0595

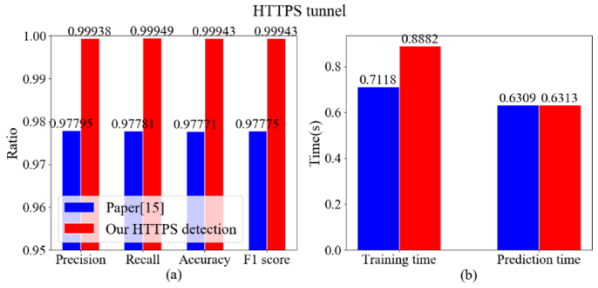

Fig. 6. Comparison of our HTTPS tunnel detection with the detection in [15].

4.2 Evaluation on DGA Filtering Rule

In addition to the above test metrics related to the machine learning model, we should also test whether the detection efficiency of the proposed detection method is improved after adding the DGA filtering rule. We only analyzed the effectiveness of the proposed DGA filtering rule by comparing the time taken before and after applying the DGA filtering rule.

Table 6. Preprocessing and prediction time of three tunnel detection models.

Detection model	Single sample preprocessing time (ms)	Single sample prediction time (ms)
DNS tunnel	10.154624649468178	0.00125437832283
HTTP tunnel	3.007999029759786	0.00162138299147

HTTPS tunnel	44.167038655595921	0.00107835463776

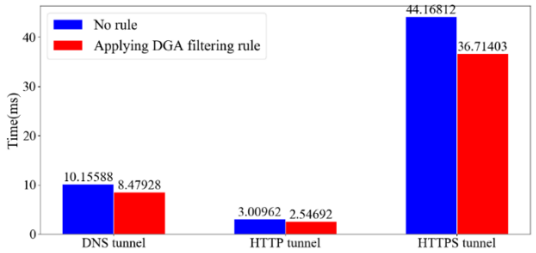

Fig. 7. Time taken before and after applying the DGA filtering rule.

Recall that applying the DGA filtering rule can detect 0.33973 black samples when the precision and false positive rate are 0.99903 and 0.000383 respectively. In other words, while meeting the requirements of high precision and low false positive rate, we can directly eliminate 0.33973 application-layer tunnels after adding the DGA filtering rule. The preprocessing time and prediction time of each session of DNS, HTTP and HTTPS tunnel detection models is shown in Table 6 and the DGA filtering rule takes 0.04853 ms (millisecond) to compute each domain name. The time consumption taken before and after applying the DGA filtering rule is shown in Fig. 7. We can see that the efficiency with regard to detection time of the proposed method for detecting DNS, HTTP and HTTPS tunnels is improved by applying the DGA filtering rule.

4.3 Further Discussion

In summary, the proposed method has the following advantages:

High efficiency. The designed DGA domain name filtering rule can effectively identify some tunnels with obvious domain name features, thereby reducing the amount of data used for machine learning and improving detection efficiency. In addition, by optimizing the feature signature-based detection method, we do not process and analyze the protocol payload data for further improving detection efficiency.

Generality. Our method can be applied to detect most tunnels based on application layer protocols.

Real-time. In this paper, a DGA filtering rule is designed to avoid the recombination of some network sessions, thus improve the response speed of tunnel detection. Of course, in order to achieve high detection accuracy, it is necessary to reassemble a network session in the step of machine learning-based detection.

High accuracy and low false positive rate. Compared with the existing work, the proposed generic feature extraction framework achieves high accuracy and low false positive rate.

Resistance against bypassing attack. The feature extraction of detection method is based on the generic feature extraction framework and extracts features from multiple perspectives. Thus, it is difficult for an attacker to spoof the detection.

Resistance against encryption issue. In the proposed method, we do not analyze protocol payload data, but only analyze the protocol header and communication behaviors. Therefore, we can deal with the network data whose payload is encrypted.

Privacy preservation. Without analyzing and processing the protocol payload data, the privacy leakage of protocol payload is avoided and the availability of this method is improved.

The shortcoming of this work is that the experimental data is collected in the campus network, i.e., the proposed method is not tested in the actual production environment.

5 Conclusion

The application-layer tunnel is one of the main means of communications for network threats in recent years. In order to detect network threats and block their communication connections, we studied the application-layer tunnel detection methods and proposed a comprehensive detection method based on rules and machine learning. We first applied the DGA domain name filtering rule to eliminate the application layer tunnels with obvious domain name features for reducing the amount of data needed by the subsequent machine learning-based detection in order to improve detection efficiency and support real-time detection. For ensuring detection accuracy, machine learning-based detection is further applied with the support of a feature extraction framework. Concretely, we constructed DNS, HTTP and HTTPS tunnel detection models as examples to evaluate our method. Experiments showed that it is a generic framework with high accuracy and low false positive rate. In the future, we plan to extend the proposed method to other types of application-layer tunnels and deploy it in practice.

Acknowledgement

This work is sponsored by the National Key Research and Development Program of China (Grant 2016YFB0800700), the National Natural Science Foundation of China (Grants 61672410 and U1536202), the Academy of Finland (Grants 308087 and 314203), the open grant of the Tactical Data Link Lab of the 20th Research Institute of China Electronics Technology Group Corporation (grant No. CLDL- 20182119), the Key Lab of Information Network Security, Ministry of Public Security (Grant C18614).

References

1. Nuojua, V., David, G., Hämäläinen T.: DNS Tunneling Detection Techniques – Classification, and Theoretical Comparison in Case of a Real APT Campaign. In: International Conference on Next Generation Wired/Wireless Networking, pp. 280-291. Springer, Cham (2017).
2. Borders, K., Prakash, A: Web tap: detecting covert web traffic. In: Proceedings of the 11th ACM conference on Computer and communications security, pp. 110-120. ACM, New York (2004).

3. Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: Detecting http tunnels with statistical mechanisms. In: 2007 IEEE International Conference on Communications, pp. 6162-6168. IEEE, Glasgow (2007).
4. Dusi, M., Crotti, M., Gringoli, F., Salgarelli, L.: Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. Computer Networks 53(1), 81-97 (2009).
5. Do, V. T., Engelstad, P., Feng, B., van Do, T.: Detection of DNS Tunneling in Mobile Networks Using Machine Learning. In: International Conference on Information Science and Applications, pp. 221-230. Springer, Singapore (2017).
6. Almusawi, A., Amintoosi, H.: DNS Tunneling detection method based on multilabel support vector machine. Security and Communication Networks 2018, (2018).
7. Qi, C., Chen, X., Xu, C., Shi, J., Liu, P.: A bigram based real time DNS tunnel detection approach. Procedia Computer Science 17, 852-860 (2013).
8. Aiello, M., Mongelli, M., Papaleo, G. DNS tunneling detection through statistical fingerprints of protocol messages and machine learning. International Journal of Communication Systems 28(14), 1987-2002 (2015).
9. Liu, J., Li, S., Zhang, Y., Xiao, J., Chang, P., Peng, C.: Detecting DNS Tunnel through Binary-Classification Based on Behavior Features. In: IEEE Trustcom/BigDataSE/ICESS, pp. 339-346. IEEE, Sydney (2017).
10. Ding, Y. J., Cai, W. D.: A method for HTTP-tunnel detection based on statistical features of traffic. In: 2011 IEEE 3rd International Conference on Communication Software and Networks, pp. 247-250. IEEE, Xi'an (2011).
11. Piraisoody, G., Huang, C., Nandy, B., Seddigh, N.: Classification of applications in HTTP tunnels. In: 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet), pp. 67-74. IEEE, San Francisco (2013).
12. Li, S., Yun, X., Zhang, Y.: Anomaly-based model for detecting HTTP-tunnel traffic using network behavior analysis. High Technology Letters 20(1), 63-69 (2014).
13. Mujtaba, G., Parish, D. J.: Detection of applications within encrypted tunnels using packet size distributions. In: 2009 International Conference for Internet Technology and Secured Transactions (ICITST), pp. 1-6. IEEE, London (2009).
14. Wang, F., Huang, L., Chen, Z., Miao, H., Yang, W.: A Novel Web Tunnel Detection Method Based on Protocol Behaviors. In: International Conference on Security and Privacy in Communication Systems, pp. 234-251. Springer, Cham (2013).
15. Allard, F., Dubois, R., Gompel, P., Morel, M.: Tunneling activities detection using machine learning techniques. Journal of Telecommunications and Information Technology 2011(1), 37-42 (2011).
16. Buczak, A. L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials 18(2), 1153-1176 (2016).
17. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E. S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials 21(1), 686-728 (2018).
18. Wang, T. S., Lin, H. T., Cheng, W. T., Chen, C. Y.: DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. Computers & Security 64, 1-15 (2017).
19. Khehra, G., Sofat, S.: BotScoop: Scalable Detection of DGA Based Botnets Using DNS Traffic. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-6. IEEE, Bangalore (2018).
20. Alexa Top 1 Million Sites, http://www.alexa.com/topsites, last accessed 2019/01/20.
21. 360 Netlab Open Data DGA, https://data.netlab.360.com/dga/, last accessed 2019/01/20.