doi:10.3772/j. issn. 1006-6748. 2014. 01. 010

Anomaly-based model for detecting HTTP-tunnel traffic using network behavior analysis①

Li Shicong (李世淙)* ** ***, Yun Xiaochun* **, Zhang Yongzheng②**
(*Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
(**Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, P. R. China)
(***University of Chinese Academy of Science, Beijing 100190, P. R. China)

Abstract

Increasing time-spent online has amplified users' exposure to the threat of information leakage. Although existing security systems (such as firewalls and intrusion detection systems) can satisfy most of the security requirements of network administrators, they are not suitable for detecting the activities of applying the HTTP-tunnel technique to steal users' private information. This paper focuses on a network behavior-based method to address the limitations of the existing protection systems. At first, it analyzes the normal network behavior pattern over HTTP traffic and select four features. Then, it presents an anomaly-based detection model that applies a hierarchical clustering technique and a scoring mechanism. It also uses real-world data to validate that the selected features are useful. The experiments have demonstrated that the model could achieve over 93% hit-rate with only about 3% false-positive rate. It is regarded confidently that the approach is a complementary technique to the existing security systems.

Key words: network security, anomaly detection model, hierarchical clustering, HTTP-tunnel

0 Introduction

Information leakage has been recognized as one of the most serious problems in network security, and is the primary concern of network researchers and administrators. Different from other types of network attacks, worm scanning and distributed denial of service involves stealing information quietly with no intent to impact the network in obvious ways. To avoid being detected, stolen information is transmitted via convert channels and disguised as the protocol traffic allowed crossing the edge of a local network. A number of Trojans and other types of spyware programs are fitted with an HTTP-tunnel mechanism to steal information, which have made the detection of stolen information extremely difficult.

Due to the growing concern of network security, systems such as firewalls and intrusion detection systems have been widely deployed to protect local networks.

The firewalls are commonly installed to implement predefined security policies and to block inbound and outbound traffic by checking whether the protocols, communication ports and the information carried conform to the security requirements of network managers. Many work focusing on firewall have been carried out. Gupta, et al.[1] develop efficient data structures to speed up the checking of firewall rules when a new packet arrives. Bartal, et al.[2] developed a high-level specification and simple model definition language used to specify firewall policies. Cheng and Liu[3,4] dedicated their work to finding collaborative firewall policies in virtual private network (VPN), intending to enforce firewall policies on encrypted VPN tunnels. However, despite these advances in research, the usefulness of firewall has been impacted by the tunneling techniques. Further research has involved developing a set of power techniques to tunnel specified application protocols into others (such as HTTP, DNS, and POP3), thus making it impractical to block traffic over them[5]. For example, if hackers tunnel steal information into HTTP protocols, it can be effectively manipulated to circumvent the block rules.

Intrusion detection systems generally apply a signature-based strategy to monitor the whole network. In

① Supported by the National Natural Science Foundation of China (No. 61070185, 61003261) and the Knowledge Innovation Program of the Chinese Academy of Sciences (No. XDA06030200).

② To whom correspondence should be addressed. E-mail: zhangyongzheng@ iie. ac. cn

Received on Sep. 7, 2012

this approach, the system utilizes deep packet inspection (DPI) techniques to extract the application-level signature for detection of information leakage. Intrusion detection systems, such as BRO and Snort, have applied DPI techniques to detect abnormal traffic[6,7]. Borders, et al.[8] proposed a tool called Web Tap to detect attempts at sending information through HTTP tunnels. Web Tap, however, involves analyzing the content of the application layer. Borders, et al.[9] presented an approach for detecting information leakage, using the insight that most network traffic patterns are repeated. Rossow, et al.[10] proposed a network behavior analysis environment called Sandnet which complements existing systems by focusing on network traffic analysis. Although signature-based intrusion detection could result in a high rate of accuracy, this approach suffers from the following limitations: (1) The detection approach is able to discover known patterns of information only while it is inefficient in detecting novel patterns; (2) with the increase of the signatures, approaches become computationally more expensive; and (3) the detection approach may not be capable of addressing even simple encryption means. Considering these shortcomings, the existing network protection systems are not considered sufficiently effective to address tunneling techniques.

In this paper, we present an anomaly-based detection model based on network behavior analysis and capable of addressing the limitations inherent to existing protection systems. Network behavior analysis is widely used to classify application layer traffic[11] and to detect the Malware, which can consist of a number of distributed agents[12]. Our model consists of the following three components: 1) connection features extraction, 2) normal behavior modeling, and 3) anomaly detection. Then, we have applied several real-world network traces to train and evaluate our model.

To summarize, there are several significant contributions stemming from our research. First, we have analyzed several real traces and extracted more stable and sustainable features to better describe the user's network behavior over HTTP protocol. So, our model is able to adapt to network changes. Second, while our work is based on normal network behavior and establishes the normal behavior patterns, it is also suitable for detecting abnormal network behavior without considering the inconsistent and multiple anomaly behavior patterns. Third, with the emergence of encryption means detection techniques based on the application layer alone are becoming insufficient in the discovery of malicious network intrusion. However, our approach invests features from both the network and transport layers, and is therefore independent of the application layer.

The rest of this paper is structured as follows: Section 1 presents the model architecture and features selected to describe normal behavior over HTTP traffic; Section 2 illustrates the mechanism of our anomaly detection model based on network behavior analysis; the results of HTTP-tunnel traffic detection are exhibited in Section 3; finally, Section 4 concludes our research.

1 Model architecture and features selection

1.1 Overview of model architecture

The goal of our detection model is to discover the abnormal HTTP traffic generated by HTTP tunnels. According to the characteristics of normal HTTP traffic, a baseline of normal behavior is established to detect deviating HTTP-tunnel traffic. Here, an overview of the model architecture and each component's functions are presented. The model is composed of three components, as shown in Fig. 1: connection features extraction, normal behavior modeling, and anomaly detection.

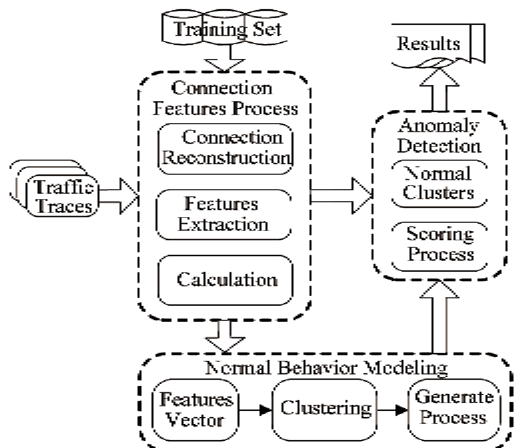


Fig. 1 The architecture of detection model

The input of the model is the raw packets collected from the edge of the local network. In the connection features extraction, three processes are applied to deal with the raw data trace and generate connection feature vectors. At first, TCP connections are reconstructed when packets are captured. With our work focusing solely on TCP-based connections, the specified features analyzed manually over HTTP normal traffic will be calculated. These features are efficient enough to describe the HTTP normal behavior. Finally, the features of a TCP connection are expressed by a vector containing four elements (to be discussed later in the paper).

During the phase of normal behavior modeling, the feature vectors are used to establish normal behavior clusters. During this process, hierarchical clustering technique is used to generate normal clusters and then log-likelihood methods are applied to measure the distances between clusters. The hierarchical clustering technique is a scalable cluster analysis method able to handle both continuous and categorical variables[13].

The clusters are used in the process of anomaly detection, and a scoring mechanism is applied to uncover the HTTP-tunnel traffic. An anomaly score is assigned to each testing instance depending on the degree of the deviation from the instance of its closest cluster. Then, the anomaly instances are identified by comparing scores with the predefined threshold (specified as three in this paper).

1.2 Features selection

Features selection is crucial to the detection process. Many connection features are available, however, not all features can appropriately describe the normal behavior over HTTP traffic. In this paper, features at the transport level are extracted, making it depend largely on reconstructing the TCP connections. Four aspects closely related to HTTP normal behavior are considered (a detailed discussion of which will take place in Section 3).

These four aspects of HTTP normal behavior are: 1) users have a tendency to surf the Internet while at work. Thus, resulting HTTP traffic is generated at roughly a set time each day (for instance, from 8:00 am to 5:00 pm); 2) when we contemplate the design of webpages, we understand that a single webpage consists of many small entities. Therefore, there are a series of HTTP requests issued to obtain these small entities, 3) although the webpage entities are small, their sizes are still larger than the size of the requests. In other words, the numbers of request packets are usually much less than the responses, and 4) when hackers receive stolen information from malicious programs, analyzing time is required to comprehend the information. Thus, there is an obvious fact that the analyzing time is much longer than normal packet interval time.

These four features we selected are illustrated as follows:

① init-time: the initialization time of the TCP connection, discretized and arranged to six different ranges.

② in-out-packets: the compare of request and response packets numbers. If the number of response is larger than that of requests, its value equals 1. Otherwise the value equals 0.

③ duration: the duration of TCP connection over HTTP traffic.

④ mean-interval: the average of packet interval time for a connection over HTTP traffic.

2 The methodology of modeling

The anomaly-based detection procedure searches for abnormal HTTP connections based on their deviations from normal clusters. The procedure is divided into two stages: clustering and scoring.

2.1 Clustering

The cluster step takes sub-clusters (excluding the outliers) resulting from the pre-cluster step as input, and then arranges them into the desired number of clusters. Since the number of sub-clusters is much less than that of the original instances, traditional clustering techniques can be used effectively. In this stage, a hierarchical clustering algorithm is applied to build normal clusters. Our reasons for using this unsupervised machine learning technique are as follows: First, we focus on the HTTP normal behavior only, meaning that the abnormal instances are not necessary to help train the model. Due to the unsupervised nature of this mechanism, we are also free from labeling the instance and generated clusters. Second, the hierarchical clustering algorithm is a scalable cluster analysis method designed to deal with very large data sets. Third, this algorithm is able to handle both continuous and categorical variables while requiring only once data traverse.

In clustering process, a cluster is characterized by some statistics including the number of the instances, mean, and variance of each range field. Additionally, the cluster counts for each category of each symbolic field. At the beginning of the process, each instance is defined as an initial cluster. All clusters are compared, and the two clusters with the smallest distance between them are merged into a single cluster. Then, the new group of clusters is compared, the closest pair is merged, and the process continues until all clusters have been merged. The explicit algorithm is outlined in **Algorithm 1.**

```
Algorithm 1: Hierarchical clustering algorithm
Input: connection instances: $I = \{I_1, I_2, \cdots, I_n\}$
Output: clusters: $C = \{C_1, C_2, \cdots, C_k\}$
1.  Specify the number of output clusters $k$;
2.  Define a variable $j$ as the current number of clusters, $j = n$, $C_i = \{N_i, S_{Ai}, S_{Ai}^2, N_{Bi}\}$, where $N_i$ is the number of instances in $C_i$, $S_{Ai}$ is the sum of the continuous variables' values, $S_{Ai}^2$ is the sum of squared continuous variables' values, $N_{Bi}$ is the number of categorical variables;
```

```
3. If $j = k$, then output $C = \{C_1, C_2, \cdots, C_k\}$;
4. For each instance, calculate distance $d(C_e, C_f)$ between $C_e$ and $C_f$, and select the smallest one;
5. Merge $C_e$ and $C_f$ into a new cluster, $C_{<e,f>} = \{N_e + N_f, S_{Ae} + S_{Af}, S_{Ae}^2 + S_{Af}^2, N_{Be} + N_{Bf}\}$;
6. Delete $C_e$ and $C_f$, $j = j - 1$, repeat 3,4,5,6.
```

The hierarchical clustering algorithm uses a log-likelihood distance measure to handle both continuous and categorical variables, and is regarded as a probability-based distance. The distance between clusters k and s is defined as

$$d(k,s) = \eta_k + \eta_s - \eta_{<k,s>} \tag{1}$$

where

$$\eta_g = -N_g\left(\sum_{m=1}^{M^\alpha} \frac{1}{2}\ln(S_m^2 + S_{gm}^2) + \sum_{m=1}^{M^\beta} E_{gm}\right) \tag{2}$$

and

$$E_{gm} = -\sum_{l=1}^{L_m} \frac{N_{gml}}{N_g} \ln \frac{N_{gml}}{N_g} \tag{3}$$

Table 1 defines the variables used in the above expressions.

Table 1 Clustering variables

Variable	Implication
M^α	The number of continuous variables
M^β	The number of categorical variables
L_m	The number of categories of the m^{th} categorical variable
N_g	The number of instances of cluster g
N_{gml}	The number of instances whose m^{th} categorical variable values are l in cluster g
S_m^2	The variance of all m^{th} continues variables
S_{gm}^2	The variance of all m^{th} continues variables in cluster g
$< k,s >$	The new cluster created by merging clusters k and s

2.2 Scoring

An important factor for any anomaly detection is the method by which anomalies are reported. Scoring techniques assign an anomaly score to each instance in the test data depending on the degree to which that instance is considered an anomaly[14]. In our work, we scored all detected instances and identified them as anomalies by comparing their scores with the predefined threshold of three. Initially, scoring data should contain feature variables in the training data. Moreover, the format of the variables in the scoring data should be the same as those in the training data set at the clustering stage. Here, each instance of testing set is described as a feature vector and the clusters from the clustering stage are applied to the scoring process. For each cluster, the average log-likelihood distance is calculated for all its instances and the testing instances are assigned to their closest cluster, respectively. Finally, the anomaly score for each testing instance is calculated to identify whether the instance is anomaly or not. The algorithm is outlined explicitly in **Algorithm 2.**

```
Algorithm 2: Scoring algorithm
Input: a testing set: $I = \{I_1, I_2, \cdots, I_m\}$, clusters: $C = \{C_1, C_2, \cdots, C_k\}$
Output: an anomaly set: $I^a = \{I_1^a, I_2^a, \cdots, I_f^a\}$
1. Specify the value of threshold $T$;
2. Each $I_i \in I$ is generally characterized as $V_i = \{V_{1i}, V_{2i}, \cdots, V_{ji}\}$, where $V_i$ is the feature vector of $I_i$ and $j$ is the number of variables of $I_i$;
3. Introduce $C = \{C_1, C_2, \cdots, C_k\}$, where $C$ is a group of clusters from the clustering stage;
4. Calculate $ave_h = \sum_{t=1}^{n_h} d(I_t, C_h)/n_h$ based on (1), where $I_t$ is an instance of cluster $C_h$, and $n_h$ is the number of instances in $C_h$;
5. Assign each $I_i \in I$ to its closest cluster;
6. Calculate $dc_{hi} = d(I_i, C_h)$ using (1), $C_h$ is the closest cluster to $I_i$;
7. For each $I_i \in I$, if $\frac{dc_{hi}}{ave_h} > T$ then assign $I_i$ to $I^a$;
8. Output $I^a = \{I_1^a, I_2^a, \cdots, I_f^a\}$.
```

3 Experimental analysis and evaluation

3.1 Empirical traces

To evaluate our detection model, two empirical packet traces have been used as the experimental datasets. One is a publicly available packet trace collected at the edge of the University of Waikato network and named as DATA1①. One subset from this trace (July 08, 2007 at 00:00:00 to July 09, 2007 at 00:00:00) was used to establish the model and another (August

① http://www.wand.net.nz/uits

11, 2007 at 00:00:00 to August 16, 2007 at 00:00:00) was used to evaluate the model. The other trace was collected at the edge of our laboratory network and named DATA2. Similarly, one subset (April 08, 2011 at 13:00:00 to April 09, 2011 at 13:00:00) was used to establish the detection model and another (September 09, 2011 at 08:00:00 to September 14, 2011 at 10:00:00) was used to evaluat the model. The length of packets comprising the two traces is 96 bytes including TCP/IP header. The traces without anomaly was used to train the detection model, and then the model was evaluated by combining normal and abnormal HTTP traffic. The anomaly connections were generated by tunneling software[5] and three HTTP-tunnel Trojan samples② obtained publicly from the Internet.

3.2 Evaluation metrics

The performance of the detection model is demonstrated through two metrics: hit-rate and false-positive rate. Both metrics show how well the detection model detects the HTTP-tunnel traffic among the normal network traffics. The two metrics are defined as follows:

- Hit-rate (HR): Percentage of correctly identified instances among the total number of HTTP-tunnel connection instances. It is calculated as

$$HR = \frac{n_{\text{correct}}}{n_{\text{total}}} \tag{4}$$

where n_{correct} is the number of HTTP-tunnel connection instances which is identified correctly, and n_{total} is the total number of HTTP-tunnel connection instances in the testing set.

- False-positive rate (FP): Percentage of the HTTP-tunnel connection instances are identified as HTTP-tunnel connections incorrectly. It is calculated as

$$FP = \frac{n_{\text{incorrect}}}{n_{\text{indentified}}} \tag{5}$$

where $n_{\text{incorrect}}$ is the number of HTTP-tunnel connection instances identified incorrectly, and $n_{\text{identified}}$ is the number of identified HTTP-tunnel connection instances.

3.3 Statistics of HTTP connection features

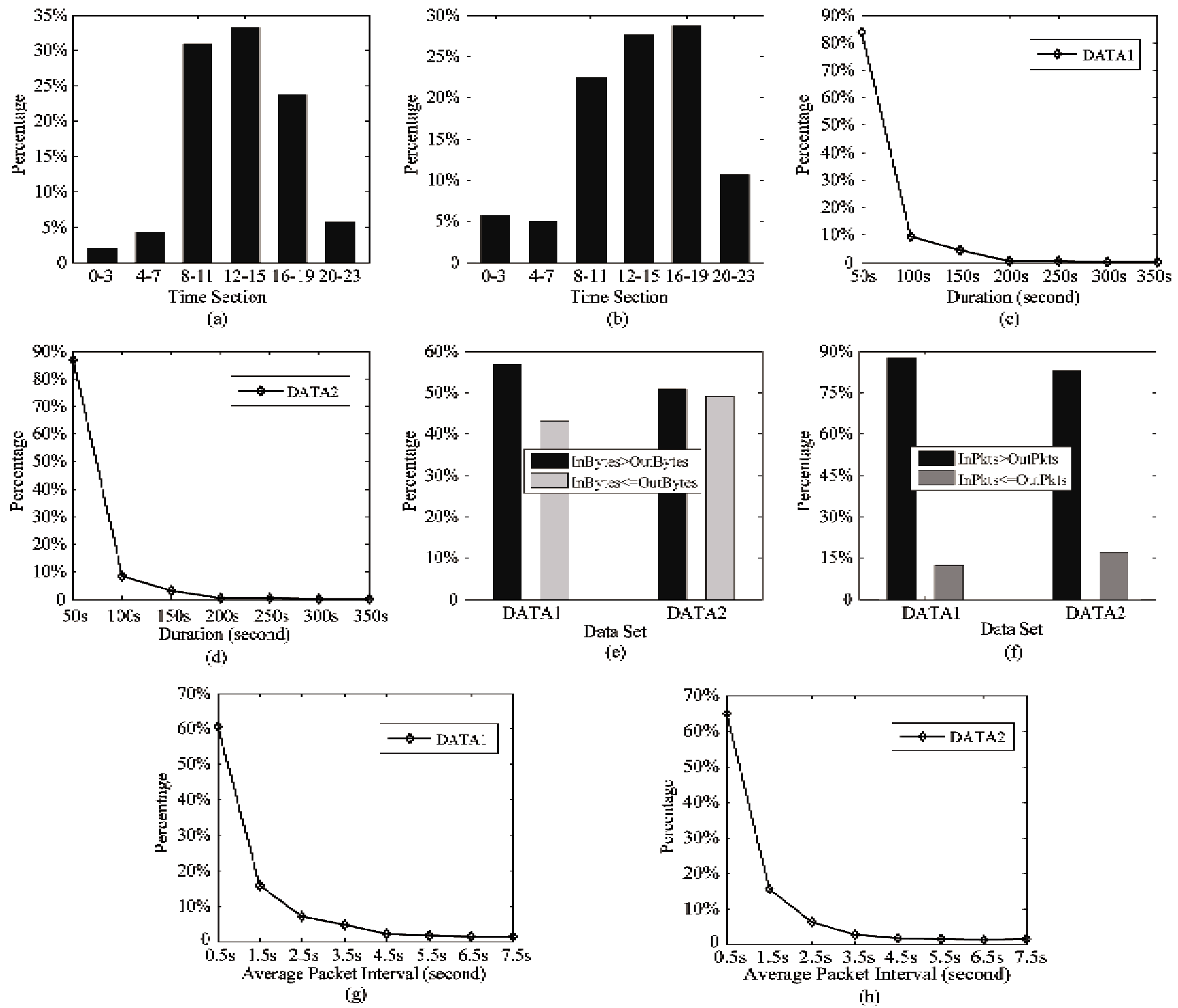
This part of our research involves performing statistical experiments for HTTP connections according to the aforementioned features. The validation of these features, which can describ appropriately the normal behaviors of HTTP connection, has been presented through our statistical analysis.

The time of day is recorded at which users usually surf the Internet and assume that people tend to follow a schedule to do their surfing at set time. Fig. 2 (a) and (b) show the validation of our assumption. A single day is divided into six sections equally to observe the number of HTTP connections appearing in these sections. From both figures we are able to demonstrate that a typical user's surfing time occurs primarily in three blocks between 8:00 a. m. and 7:00 p. m. The numbers of connections in these time blocks account for over 75% in both traces (approximately 84% in DATA1 and 77% in DATA2). This percentage could be linked to the users tending to have more rigid schedules. Even when users have different tasks and work descriptions, they still show obvious patterns in usage time.

We know that user's browsing would generate a number of TCP connections for the transmit of request and response information. For most webpages consisting of many small entities, the duration of these connections are brief. Fig. 2 (c) and (d) demonstrate this. We can see that the duration of 84% of the HTTP connections in DATA1 is less than 50 seconds, while nearly 87% in DATA2 share this same characteristic. HTTP connection durations of less than 100 seconds account for over 95% in both traces. Although the two traces are collected at different time points and different networks, the distributions in both traces share similar characteristics. Hackers would need to send out large amounts of data through HTTP connection in order to transfer files and view large directory lists. Subsequently, the connections generated by malicious programs will tend to have much longer durations than the normal HTTP connections.

In Fig. 2(e) and (f), the behavior of HTTP connections is analyzed based on the volume of traffic. Fig. 2(e) shows the comparison between inbound and outbound bytes. In the first dataset, the proportion of connections whose inbound bytes larger than outbound is about 57%, while the proportion of outbound bytes less than inbound is 43%. The difference between them is not highly notable. The difference is more highly indiscriminate in the other set. In DATA2, the inbound bytes larger than outbound proportion accounts for about 51% while the outbound less than inbound proportion is 49%. Fig. 2 (f) shows the result when over 81% of the total HTTP connections have more inbound packets than outbound in both datasets. There are about 89% connections of HTTP with more inbound packets in DATA1. In DATA2, the proportion of connections with more inbound packets is approximately 82%. In this case, the difference between inbound and outbound packets is notable. Thus, the relation between inbound and outbound packets is more effective than that in bytes to describe the HTTP normal behavior.

② http://www.heibai.net/download/class/class_318_Time_1.htm.



(a), (b) for the distribution of HTTP connections according to initial time in DATA1 and DATA2; (c), (d) for the distribution of HTTP connections according to duration in two datasets; (e), (f) for inbound bytes/packets versus outbound bytes/packets over HTTP connections; (g), (h) for the distribution of HTTP connections according to average packet interval in two datasets

Fig. 2 Overviews of statistical features in real-world traces

In section1, we mention that a hacker's analyzing time has a significant impact on the average packet interval of a connection. As such, packet interval time is very important in the detection of abnormal connections. Fig. 2(g) and (h) show the distributions of HTTP connections according to their average packet intervals. Clearly, the two traces present a similar trend. About 60% of the connections have less than a 0.5 second average packet interval in DATA1, and the proportion is more than 65% in DATA2. Interestingly, both figures show that over 80% of the connections' average packet interval is less than 2.5 second. Thus, the average packet interval can be regarded as an effective detection feature.

3.4 Results of evaluation

The number of clusters is an important parameter that should be considered. This parameter could be used to tune a detection model to obtain a high hit-rate and a low false-positive rate. For this evaluation, two evaluation sets are generated by combining normal and abnormal network traffic. Both figures (Fig. 3(a) and (b)) show the performance of the model when the numbers vary from one to five, and allow us to make the following observations. First, even with only one cluster, hit-rate reaches close to 90% with a false-positive rate of only about 3% in both datasets. Second, with the number increasing we observe the hit-rate and false-positive rate are also increasing. Third, hit-rate reaches over 90% when the number of clusters is equal to 5 in both datasets. Based on these observations, we consider one cluster to be sufficient in the detection of HTTP-tunnel connections from normal traffic in both local networks.

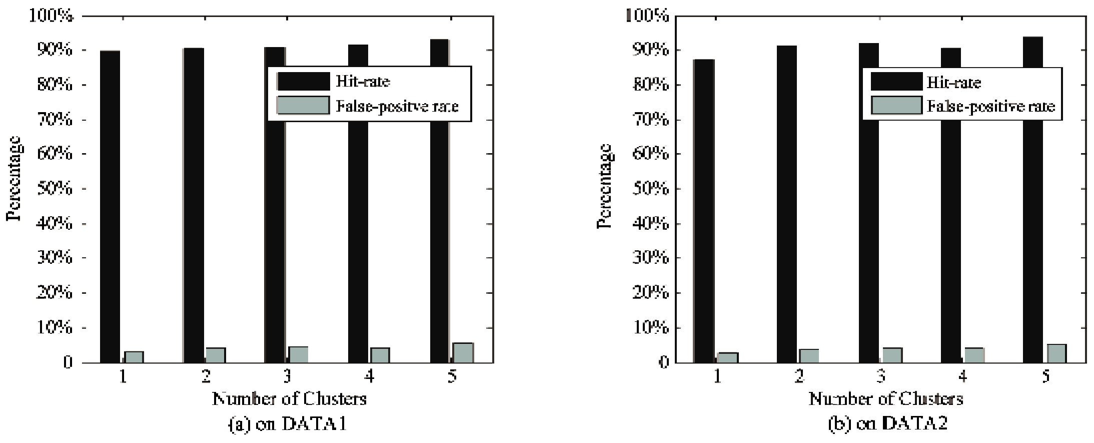


Fig. 3 Experimental performance of model

4 Conclusion

This paper specifies an important fact: existing security precaution systems are not suitable for detecting HTTP-tunnel connections among the vast amount of network traffic. An anomaly-based detection model is provided which is proven to be useful in uncovering the HTTP-tunnel connection while data presented validate that our selected features are useful. Our experiments have demonstrated that the model could achieve over 93% hit-rate with only about 3% false-positive rates.

There are many limitations which our work should be improved to address in the future. First, our model needs to observe the behavioral information of a connection as much as possible. In the current real-time context, however, we are given only partial information surrounding a connection. As such, our existing model is inefficient for real-time detection. Second, our traffic traces have been collected from only two local networks at different time. So, they can't include all the cases of HTTP-based network behavior. Third, if user uploads a large file via HTTP, this is much likely to affect the detection performance. Thus, extending our work to solve these problems is meaningful and requires future work.

References

[1] Gupta P, Mckeown N. Algorithms for packet classification. *IEEE Network*, 2001, 15(2): 24-32

[2] Bartal Y, Mayer A J, Nissim K, et al. Firmato: a novel firewall management toolkit. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, USA, 1999. 17-31

[3] Cheng J, Yang H, Wong S H, et al. Design and implementation of cross-domain cooperative firewall. In: Proceedings of the 15th IEEE International Conference on Network Protocols, Beijing, China, 2007. 284-293

[4] Liu A, Chen F. Collaborative enforcement of firewall policies in virtual private network. In: Proceedings of the 27th ACM Symposium on Principles of Distributed Computing, Toronto, Canada, 2008. 95-104

[5] Crotti M, Dusi M, Gringoli F, et al. Detecting http tunnels with statistical mechanisms. In: Proceedings of the 42th IEEE International Conference on Communications, Glasgow, Scotland, 2007. 6162-6168

[6] Paxon V. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 1999, 31 (23): 2435-2463

[7] Roesch M. Snort: lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration, Washington, USA, 1999. 229-238

[8] Borders K, Prakash A. Web Tap: detecting covert web traffic. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington DC, USA, 2004. 110-120

[9] Borders K, Prakash A. Quantifying information leaks in outbound web traffic. In: Proceeding of the 30th Symposium on Security and Privacy, Oakland, USA, 2009. 129-140

[10] Rossow C, Dietrich C, Bow H, et al. Sandnet: network traffic analysis of malicious software. In: Proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 2011. 78-88

[11] Nguyen T, Armitage G. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys & Tutorials*, 2008, 10(4): 56-76

[12] Yen T, Reiter M. Traffic aggregation for malware detection. In: Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2008. 207-227

[13] Guha S, Rastogi R, Shim K. Cure: An efficient clustering algorithm for large database. In: Proceedings of ACM SIGMOD Record, Washington, USA, 1998. 73-84

[14] Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. *ACM Computing Surveys* (*CSUR*), 2009, 41 (3): 15-94

Li Shicong, born in 1981, and is currently a Ph. D. candidate in Institute of Computing Technology, Chinese Academy of Science (CAS). He was a visiting student of the Department of National Engineering Laboratory for Information Security Technologies at Institute of Information Engineering, CAS. His researches focus on network security, network behavior analysis.