Intrusion Detection for Encrypted Web Accesses

Akira Yamada
KDDI R&D Laboratories Inc.
yamada.akira@kddilabs.jp

Yutaka Miyake
KDDI R&D Laboratories Inc.
miyake@kddilabs.jp

Keisuke Takemori
KDDI R&D Laboratories Inc.
takemori@kddilabs.jp

Ahren Studer
Carnegie Mellon University
astuder@ece.cmu.edu

Adrian Perrig
Carnegie Mellon University
adrian@ece.cmu.edu

Abstract

As various services are provided as web applications, attacks against web applications constitute a serious problem. Intrusion Detection Systems (IDSes) are one solution, however, these systems do not work effectively when the accesses are encrypted by protocols. Because the IDSes inspect the contents of a packet, it is difficult to find attacks by the current IDS. This paper presents a novel approach to anomaly detection for encrypted web accesses. This approach applies encrypted traffic analysis to intrusion detection, which analyzes contents of encrypted traffic using only data size and timing without decryption. First, the system extracts information from encrypted traffic, which is a set comprising data size and timing for each web client. Second, the accesses are distinguished based on similarity of the information and access frequencies are calculated. Finally, malicious activities are detected according to rules generated from the frequency of accesses and characteristics of HTTP traffic. The system does not extract private information or require enormous pre-operation beforehand, which are needed in conventional encrypted traffic analysis. We show that the system detects various attacks with a high degree of accuracy, adopting an actual dataset gathered at a gateway of a network and the DARPA dataset.

1 Introduction

While web-based applications are becoming common, attacks against these applications pose a serious problem. An Intrusion Detection System (IDS) is one way of dealing with such attacks. An IDS is located beside the web server and monitors the users' activities by protocol analysis and pattern matching. In other words, IDSes reconstruct HTTP headers and payload from captured packets, and identify attacks by comparing traffic to signatures of attacks. Thus the process requires the privilege of watching the entire payload of packets.

Mechanisms such as SSL (Secure Socket Layer) [1, 2] or its successor TLS (Transport Layer Security Protocol) [3] have been proposed as ways of ensuring secure communications over the Internet. These protocols make it possible to authenticate the server and client, and to safeguard the integrity and confidentiality of data. For encrypted traffic, a conventional IDS needs a deposited private key; otherwise, it needs to monitor the traffic after decryption. These conventional approaches are problematic from the perspective of key management and network configuration. However, such situations are becoming more common, because web applications that require secure communication between client and server are increasing in popularity. Therefore, the managers of web application servers are faced with the dilemma in that they want to provide secure services using SSL/TLS, but the system has become less secure due to the lack of IDS monitoring. Our main objective is to solve this problem.

In related research, traffic analysis attacks against encryption protocols have been proposed. These attacks extract covert information from encrypted traffic without decryption. Several studies [4, 5, 6, 7, 8] have reported that the type of content can be estimated, even if the traffic is encrypted by SSL, WEP or IPsec. Observing only the volume and the interval of transferred data, they can identify the type of content or destination URL with a high level of accuracy. However, the objective of these studies is to gather private information from encrypted traffic, and there are some problems in applying these methods to the direct monitoring of traffic. Because they have to compare encrypted traffic with enormous archives of unencrypted traffic, they need to gather unencrypted data before the analysis. It is difficult for an IDS to gather all of the content before because a Web server and the IDS are managed separately. Furthermore, the process consists of comparisons with every archive, and requires too much processing. Such excessive computation prevents real-time analysis for intrusion detection.

We propose a novel approach to anomaly detection for encrypted web accesses. The approach applies encrypted traffic analysis to intrusion detection. Our approach uses only data size and timing of traffic without decryption to analyze the content of encrypted traffic. First, the system extracts data size and timing information for each web client from the encrypted traffic. Second, the clients' accesses are distinguished based on similarity of the information and access frequencies are calculated. Finally, frequencies of accesses and characteristics of HTTP traffic are used to detect malicious activities.

We evaluate our system using an actual dataset gathered at a network gateway and a DARPA IDS evaluation dataset [10, 9]. We apply the algorithm to unencrypted accesses instead of encrypted traffic in this evaluation, but we account for the influence of encryption in the form of random padding for each data size. The evaluation shows that our system achieves a high degree of detection accuracy.

This paper is organized as follows. Section 2 clarifies the attacks that our system targets, and Section 3 explains the proposed system. Section 4 evaluates our system, and Section 5 discusses accuracy and performance of the proposed system. Section 6 compares the proposed system with related studies. Finally, Section 7 presents our conclusions.

2 Attacks against web applications

There are a wide range of attacks against web applications. In this section, we clarify different attack classes that the system targets. Note that the attack classification does not refer to traditional signature approaches, which compare a HTTP request string with a set of signature strings. The classification makes the target clear, but it is a more abstract class than the traditional approaches. The system detects the following attack classes: scanning attacks, scripting language vulnerabilities and buffer overflows.

2.1 Scanning attacks

Scanning attacks examine the existence and configuration of a web server or proxy server at an IP address. The attacker can obtain information about the web server and/or proxy server by using simple HTTP methods, such as GET, HEAD and OPTIONS. A directory traversal attack, which accesses the parent directory and gains information about the construction of directories and files, is also categorized as a scanning attack. The following HTTP requests are examples of scanning attacks.

```
GET http://www.qq.com/ HTTP/1.1
HEAD / HTTP/1.1
OPTIONS / HTTP/1.1
GET / HTTP/1.1
```

2.2 Scripting language vulnerabilities

Most web applications use scripting languages such as Perl or PHP. A specific version of scripts or sample code distributed with the language have vulnerabilities as they allow attackers to execute arbitrary codes. The attackers examine whether a specific script is installed or not by accessing the script on the server. Once a vulnerable script is discovered by the attacker, the attacker can compromise the server. Some examples are given below.

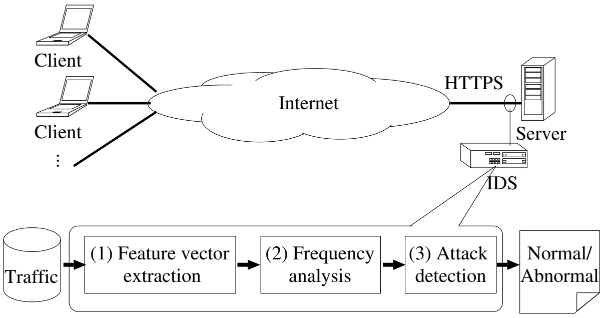

Figure 1. Proposed system configuration.

```
GET /adserver/adxmlrpc.php HTTP/1.0
GET /phpAdsNew/adxmlrpc.php HTTP/1.0
GET /phpadsnew/adxmlrpc.php HTTP/1.0
```

2.3 Buffer overflows

Attackers using a buffer overflow attack can execute arbitrary code on the web server by overwriting stack or heap memory of the process. Though usual programs check the bounds of memory accesses, unchecked memory accesses allow attackers to crash or gain control of a process by sending a larger request or argument. In the worst case, the attacker can control the web server through the vulnerability. It is possible that web applications, modules, and script languages are also vulnerable in this way. The following is an example.

```
GET /default.ida?NNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

3 Intrusion detection for encrypted web accesses

3.1 System configuration

Fig.1 shows a network configuration of the proposed system. The system is located at the gateway of the web server, and analyzes encrypted traffic without decryption. The detection has 3 components, (1) feature vector extraction, (2) frequency analysis, and (3) attack detection. First, traffic is divided into units that describe each client's activities and a feature vector, a set of parameters that describes the activity, is extracted from each unit. Timing and size of transferred data are the parameters used for encrypted traffic analysis. Second, feature vectors are categorized into clusters based on similarity metrics, which enables the frequency of each access to be analyzed. Because the sizes of encrypted data are not equivalent to plain data, it is not possible to count up each feature vector using an ordinary method. Finally, each feature vector is judged to be a legitimate or illegitimate access based on the parameters included in the feature vector and frequency of the feature vector.

3.2 Feature vector extraction

Captured traffic is divided into several units that describe each client's activities. And a feature vector is extracted

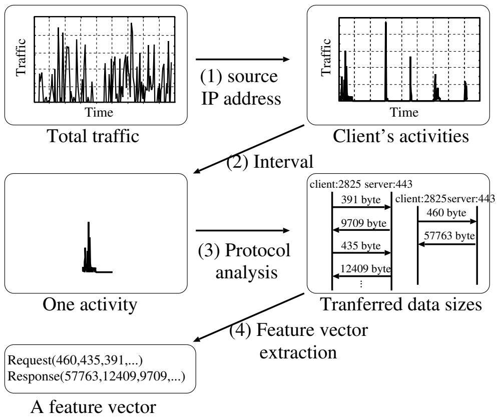

Figure 2. Feature vector extraction.

from each unit, which is a set of parameters that describe features of the activity. First, every packet is categorized into the activities of each web client, which correspond to one click in a web page. Then each activity is transformed into a feature vector, which is a set of parameters representing the activity. Web clients are distinguished based on source IP address and activities are distinguished based on interval between packets. This means that a pair of source and destination IP addresses correspond to a sequence of the client's activities, and a group of packets observed within a short period correspond to activity by that client. The feature vector extraction is as following:

1. Accesses belonging to a client are categorized based on the source IP address. We assume that a client occupies an IP address during the activities and another client does not occupy the address at the same time.

2. Activities that belong to a client are divided into each activity based on interval of the packets. We assume that if packets are observed continuously then these packets are categorized into one activity. As an example of one activity, the user clicks a hyperlink in a web page and waits for the browser response.

3. A TCP session is reconstructed from packets included in one activity and the SSL/TLS protocol is decoded from the session.

4. The feature vector is extracted as a set of volumes that are transferred at the session. The 10 largest volumes are picked up as the feature vector. Accounting for the direction of data, the feature vector consists of 20 volumes of data. The volumes are sorted in terms of size because the order of transmission does not depend on the client's activity.

Fig.2 shows the steps to extract a feature vector. The traffic is divided into activities for each client and transferred to feature vectors corresponding to the activities.

The system decodes the SSL/TLS protocol in order to estimate the volume of transferred data. Fig.3 shows the structure of a packet for SSL/TLS. The SSL/TLS protocol is constructed from the Change Cipher Spec, Handshake and Alert protocols, and these sub-protocols are built on the Record Layer protocol. The header of the Record Layer, which includes information about the type of sub-protocol and the volume of data, is not encrypted. Using a type of sub-protocol, we eliminate packets involving the negotiation of security protocol, such as key exchange or authentication. Protocol version SSL 2.0 and SSL 3.0/TSL 1.0 are slightly different. In protocol version SSL 3.0/TLS 1.0, we estimate the volume of data based on Length field in the Record Layer. In protocol version SSL 2.0, the Record Layer does not include Length field, so we estimate the volume by reassembling the TCP connection only.

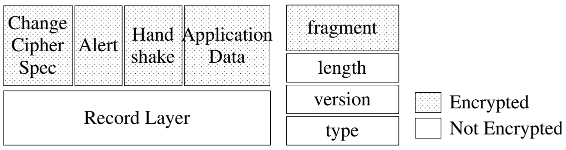

Figure 3. Structure of SSL/TLS packet.

The size of encrypted data is not equivalent to plain data because random padding is combined with the plain data, which is at most 255 bytes. SSL/TLS supports 2 types of encryption algorithm, stream cipher and block cipher. A block cipher reduces more information than a stream cipher, because a block cipher adjusts the length of data to block size. Analyzing the SSL/TLS protocol header, the system extracts approximate volumes of transferred data as the feature vector. The influence of random padding is accounted for during the evaluations.

3.3 Frequency analysis

After feature vector extraction, each feature vector is assigned a temporary ID by which the system distinguishes these activities. Because of the random padding, it is impossible to count the number of feature vectors directly. The system categorizes the feature vectors into certain groups of vectors based on similarity metric, which is Euclid distance. The influence of random padding is removed by the process; it means that the system treats similar feature vectors as a unique access. The identification process enables the system to analyze frequency of access. The algorithm is as follows.

Now N feature vectors $\boldsymbol{x_i}(i = 1, 2, \ldots, N)$ are categorized into $L(< N)$ subgroups. Each group has $ID_\ell(\ell = 1, \ldots, L)$. $N + 1$ th feature vector $\boldsymbol{x_{N+1}}$ is assigned an ID by following step. The set of vectors and the number of vectors assigned ID_ℓ are S_ℓ and N_ℓ, respectively. The average feature vector is defined as (1), which is the average size of feature vectors included in an ID_ℓ.

$$\boldsymbol{X_\ell} = \frac{1}{N_\ell} \sum_{S_\ell} \boldsymbol{x_i} \tag{1}$$

1. As the similarity metric, calculate Euclid distances between $\boldsymbol{x_{N+1}}$ and $\boldsymbol{X_\ell}$, and select minimum value $E_{\min}$

from the distances.

$$E_\ell = \mathrm{euclid}(\boldsymbol{x_{N+1}}\,, \boldsymbol{X_\ell}) \tag{2}$$
$$E_{\min} = \min_\ell [E_\ell] \tag{3}$$

2. If $E_{\min}$ is less than threshold Th_0 then $ID_{\min}$ is assigned to $\boldsymbol{x_{N+1}}$, where $ID_{\min}$ corresponds to $\boldsymbol{X_{\min}}$ and $E_{\min}$. The average feature vector and the number of feature vectors are updated as (4) and (5).

$$\boldsymbol{X_{\min}} \leftarrow \frac{1}{N_{\min}+1}(N_{\min} \times \boldsymbol{X_{\min}} + \boldsymbol{x_{N+1}}) \tag{4}$$
$$N_{\min} \leftarrow N_{\min} + 1 \tag{5}$$

3. If $E_{\min}$ is equivalent to or more than Th_0 then ID_{L+1} is assigned to $\boldsymbol{x_{N+1}}$ as a new ID. The number of the feature vectors N_{L+1} and average feature vector $\boldsymbol{X_{L+1}}$ are (6)(7).

$$N_{L+1} \quad \leftarrow \quad 1 \tag{6}$$
$$\boldsymbol{X_{L+1}} \quad \leftarrow \quad \boldsymbol{x_{N+1}} \tag{7}$$

Once IDs are assigned, the system can count up the number of times each activity occurs and the number of times a series of activities occur. Because the system regards events that occur less frequently as attacks, the number of IDs or transition between IDs has to be updated with each new activity. The accesses to the web server are continuous, and the output of the previous access is also continuous. The following algorithm counts up the number of IDs continuously. The detection of attacks is described in the Section 3.4.

Access identification outputs a sequence of M IDs, $ID^j(j = 1, 2, \ldots, M)$ where $ID^j \in \{ID_\ell\}$. M is the number of activities observed up to this time, and ID^M is the latest ID. The next ID, ID^{M+1}, updates counters using the following algorithm. The algorithm uses 2 tables T_1 and T_2, which store counters for accesses and transitions, respectively.

1. If the entry corresponding to ID^{M+1} does not exist in table T_1, then register (ID^{M+1}, c_{M+1}) to the table T_1 and set the counter ($c_{M+1} \leftarrow 1$).

2. If the entry corresponding to ID^{M+1} exists in table T_1, then read the entry (ID_{M+1}, c_{M+1}) from the table and increment the counter($c_{M+1} \leftarrow c_{M+1} + 1$).

3. When ID^{M+1} appears after ID^M, concatenate these IDs as a transition between accesses, $t_M = \{ID_M | ID_{M+1}\}$. If the entry corresponding to t_M does not exist in table T_2, then registers (t_M, c_M) to the table and set the counter ($c_M \leftarrow 1$).

4. If the entry corresponding to the transition t_M exists in table T_2, then read the entry (t_M, c_M) from the table and increment the counter ($c_m \leftarrow c_m + 1$). The algorithm is also able to handle transitions with more than 2 IDs.

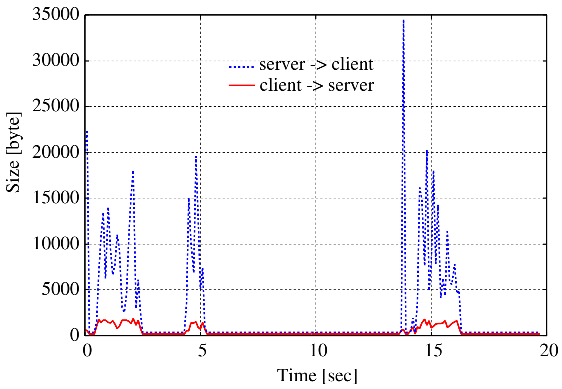

Figure 4. Transferred and received sizes in web access.

3.4 Attack detection

We assume that a statistically rare event and an event that differs from typical HTTP access behaviors are potentially malicious. The system decides whether an access is legitimate or not based on the access frequency and the characteristics of HTTP accesses. The algorithm in Section 3.3 calculates the frequency of each access. The feature vector provides the access characteristics.

We compare an observed access with the typical pattern of web accesses. The approach is similar to conventional anomaly techniques, but available parameters are limited in encrypted traffic analysis. For example, the size of a Request-line and type of method are hidden in the total size of the transferred data.

Characteristically, in a typical web access, a client sends requests consisting of a small number of bytes and receives responses consisting of a large number of bytes. Fig.4 is an example of web access that shows the transferred and received volumes. Usually a client requests a specific resource from a web server and the server responds by providing the resource, so a large-sized request or a small-sized response are abnormal. We employ two parameters, maximum size of request Th_{req} and maximum size of response Th_{res} to describe the typical pattern of web accesses.

The thresholds Th_{req} and Th_{res} correspond to the attacks listed in Section 2. Typical Scanning attacks request simple HTTP methods and only receive response codes. The responses are smaller than regular contents, even if the requests are similar to legitimate requests. An attack using script language based vulnerabilities sends a small-sized request and receives a large-sized response, which is characteristic of typical HTTP accesses. However, this behavior is only observed if the attack is succeeds. If the server is not vulnerable, the server responses should include a small message with an error status. Buffer overflow attacks have to send a larger-sized request than legitimate requests for overflowing the vulnerable buffer.

In the frequency analysis, we regard statistically rare events as potentially abnormal. Though worm infections are frequently observed, there are more legitimate accesses

Table 1. Datasets.

		MByte	Request	Instance
Actual dataset	Normal	749.8	81,386	11,977
	Abnormal	1.9	499	463
DARPA dataset	Normal	2666.8	428,630	56,261
	Abnormal	172.0	2,933	481

observed on a usual web server. After HTTP characteristic based detection, the frequency of activities is evaluated. Only rules generated by the size of request/response cause a large number of false positives. To reduce false positives, the system eliminates an alert if the event is common for the web server. We introduce the minimum threshold for events, Th_{freq}, as a parameter, and regard activities that occur more than the threshold as normal.

4 Evaluation of detection accuracy

4.1 Overview of evaluation

For the evaluation, we used 2 datasets, packets collected at a LAN gateway, and the DARPA IDS evaluation dataset. The datasets are reconstructed into legitimate and illegitimate/attack accesses. We use HTTP instead of HTTPS encrypted traffic because there are few instances of attacks on HTTPS, and an unencrypted attack can also exploit a web applications via SSL. We account for the influence of encryption through random padding, which is the most stringent condition in the protocol specifications. 0 to 255 bytes of random padding were added to each size of transferred data to make the size multiple of the block size for the encryption algorithm.

4.2 Dataset

The evaluation adopts 2 datasets, the actual dataset gathered at a network gateway and the DARPA IDS evaluation dataset. The actual dataset consists of accesses to external web sites and attacks collected by a honey pot. It is obvious that the accesses from the LAN are legitimate activities and any accesses to the honey pot are malicious. Note that accesses from a LAN are TCP connections initiated from IP addresses that belong to a private address. We chose a dynamic web site to evaluate the system, as this site provides a social networking service and possesses functions to submit an article and comment on the article. Usually, attackers target these functions of a web application, which are implemented as CGI and provide dynamic content.

The DARPA IDS evaluation dataset consists of PCAP formatted files that represent 5 weeks of traffic. We chose the files for weeks 4 and 5, because the detailed list describes attack instances that are restricted to weeks 4 and 5. While the attacks occurred against several protocols, we only use the HTTP attacks and any attacks observed on port 80. Tab.1 shows details of the datasets. Request means the number of HTTP requests and an Instance means the number of activities that the feature vector extraction outputs.

Table 2. Attacks in actual dataset.

Attack	Description
HTTP scan	Scan for existence of a web server
Proxy scan	Scan for existence of a proxy server
(CVE-2005-1921)	Multiple PHP XML-RPC implementations vulnerable to code injection
(MS03-51)	Buffer Overrun in Microsoft FrontPage Server Extensions Could Allow Code Execution
(MS04-007)	ASN.1 Vulnerability Could Allow Code Execution

Table 3. Attacks in DARPA dataset.

Attack	Description
Apache2	Denial of service attack against an apache web server
Back	Denial of service attack against an apache web server
Phf	Remote execution attack against badly written CGI script
Crashiis	Denial of service attack against an IIS web server
Mscan	Vulnerability scanner for windows NT
Ntinfo	Vulnerability scanner for network IP addresses

4.3 Attack instances

Attacks included in the dataset are listed in Tab.2 and 3. Attacks in the actual dataset are gathered by the honeypot, so the attack instances do not successfully compromise the victims. On the other hand, some attacks in the DARPA dataset successfully compromised the victims. Some attackers steal sensitive information, such as password files, from the server. HTTP scan, Proxy scan, NTinfo and Mscan are categorized as scanning attacks. (CVE-2005-1921) and Phf are categorized as attacks against scripting language vulnerabilities. (MS03-51), (MS04-007), Apache2 and Back are categorized as a buffer overflow attacks.

4.4 Error rate

Fig. 5 shows the error rate for the actual dataset and the DARPA dataset. Two thresholds, Th_{req} and Th_{res}, are evaluated; these are the maximum size of a request and the maximum size of a response respectively. In these evaluations, random padding is not included. The results from the actual dataset are presented in the top graph. The solid line corresponds to the conditions with a threshold of request size, Th_{req}, fixed at 3,000 and response size, Th_{res}, ranging from from 0 to 100,000. The broken line corresponds to Th_{res} fixed at 400 and Th_{req} ranging from 0 to 100,000. At $Th_{req} = 3000$ and $Th_{res} = 400$, the error rate is minimum. A high detection rate is achieved with a low false positive rate.

The bottom graph in Figure 5 contains the results from the evaluation of our system on the DARPA dataset. The solid line corresponds to a request threshold, Th_{req}, fixed at 800 and a response threshold, Th_{res}, from 0 to 100,000. The broken line corresponds to Th_{res} fixed at 2,000 and Th_{req} from 0 to 100,000. The error rate is at a minimum at $Th_{req} = 800$ and $Th_{res} = 400$, but the detection rate is lower here than for the actual dataset.

Fig.6 shows the error rate under the condition that random padding is introduced to the actual dataset. There is

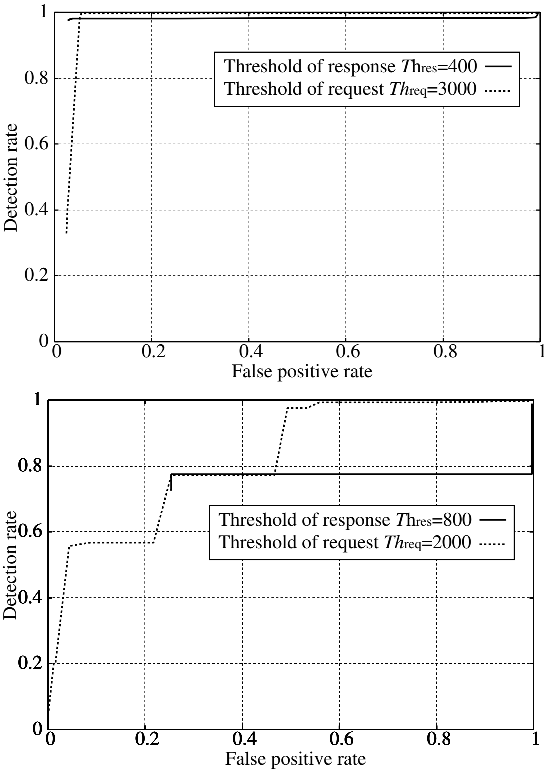

Figure 5. Detection rate for each dataset.

not much difference between the original results in Figure 5 and the results with padding in Figure 6.

Fig.7 shows the error rates of the actual dataset for varying values of the minimum frequency threshold, Th_{freq}, which is the threshold used for frequency analysis. In the first graph, threshold Th_{req} and Th_{res} are fixed at 3000 and 400, in the second graph, the thresholds are fixed at 800 and 2000. As Th_{freq} increases, FPR and FNR approach the original error rates. Though the minimum FNR and FPR are limited, FPR and FNR can be controlled by Th_{freq}. In particular, frequently occurring false alarms are eliminated by using frequency analysis. It is possible that frequency analysis could eliminate true alarms. Though all true alarms are reduced to a single alarm, the first alarm is not deleted. Both true and false alarms are detected one or more times even if the alarms are deleted due to frequency analysis.

5 Discussion

5.1 Detection accuracy

In the actual dataset, the proposed algorithm detects the attacks with low false positive and false negatives rates. Actual instances of scanning, script, and buffer overflow attacks are successfully distinguished from legitimate accesses with a high degree of accuracy. The proposed algorithm captures the characteristics of an error response that the web server sends, namely, a small-sized response.

In the DARPA dataset, the proposed algorithm fails to detect attack instances with a small-sized request and a

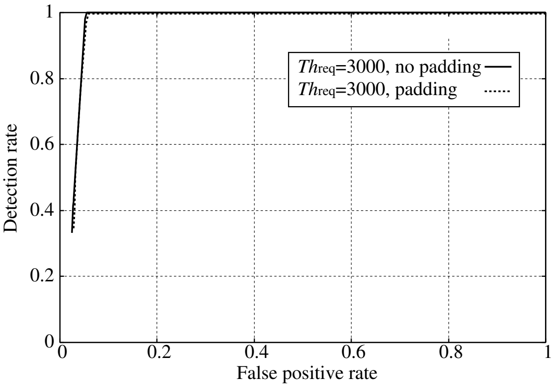

Figure 6. Detection rate for random padded dataset.

large-sized response. It means that the system can not detect attacks that are similar to legitimate accesses. In the case of the DARPA dataset, the attacker sends small-sized exploit code and gets a password file from the web server without any other scanning activity. However, in a typical scenario, an attacker would use scanning activities to carry out the attacks. To find an unknown vulnerability in a web application server, the attacker needs to try several scanning requests. The proposed algorithm would reveal unknown attacks by the related scanning activities, even if the unknown attacks are similar to legitimate accesses. Under such conditions, the approach would not be susceptible to high false positive/negative rates. In future work, we plan to evaluate our system using other traffic datasets and attack instances collected from actual networks.

5.2 Performance

For real time intrusion detection, performance is important. During the proposed access identification, the number of IDs grows dramatically and causes significant consumption of memory and the need for considerable calculation power. To address this issue the algorithms were extended to use Least Recently Used (LRU) ID management. LRU enables the algorithms to work continuously while using limited memory with old entries deleted automatically. In a web site, the old contents are updated, or deleted and never accessed. However, the system cannot recognize the content update, so entries related to the old content are needlessly retained permanently. If the old entries are deleted automatically using LRU, the system is able to perform continuous analysis with limited memory. It is true that the accuracy of the algorithms is reduced in the additional process, but high performance and continuous analysis without the need for maintenance is a significant benefit for network operation.

6 Related work

6.1 Inspection for encrypted traffic

In [11], conventional inspection techniques for encrypted web accesses are summarized. There are 3 approaches to

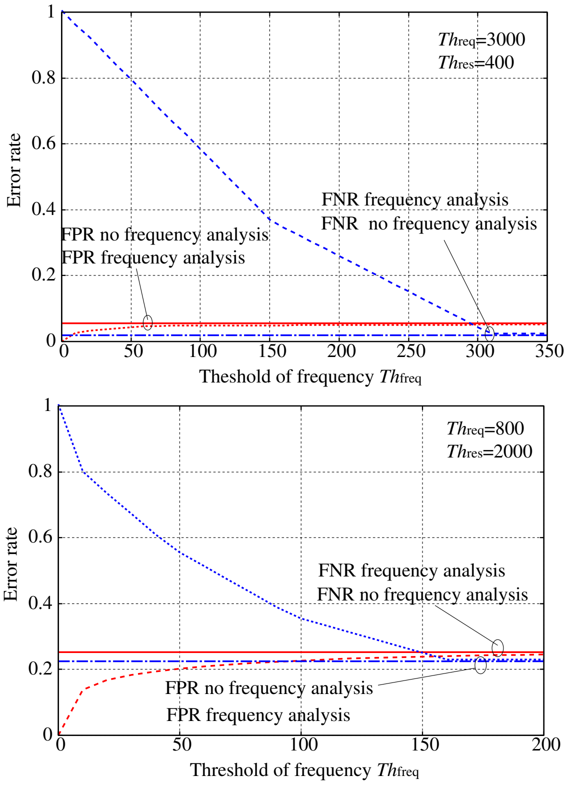

Figure 7. Result of frequency analysis.

the inspection of encrypted traffic; terminate the SSL connection at the IDS, have the IDS passively decrypt the data using a copy of the server's private key, and collocate the IDS on the web server. The network configurations are shown in Fig.8. There are several drawbacks in the above configurations. The first two configurations have a problem in managing the private key for SSL. There is a greater risk of the key being stolen as the IDS has to have the private key It is also troublesome that to terminate the SSL connection at the IDS requires reconfiguration of the network. The second technique, the IDS passively decrypts SSL data, is only applicable to SSL using the RSA key exchange. If the cipherspec uses Diffie-Hellman key exchange, the method does not work well. An IDS collocated on the web server must use some of the resources that would otherwise be available for the web server; CPU power, hard drive space, memory and so forth. As described above, conventional monitoring methods have problems inspecting encrypted traffic.

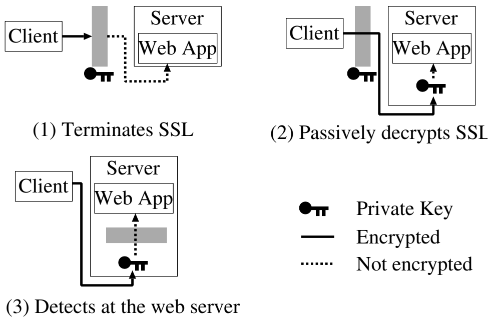

Figure 8. Conventional approaches to the inspection of encrypted web accesses.

Yasinsac et al. proposed a method for detecting intrusions for encrypted traffic [12, 13], but the objectives are different from those contained in our proposal. They also address the problem of intrusion detection for encrypted traffic, but they specialize in attacks against security protocols, for example, replay attacks and parallel session attacks. We do not target attacks against security protocols but attacks encrypted by security protocols.

6.2 Encrypted traffic analysis

Traffic analysis attacks are addressed in the specification of SSL/TLS [3]. The protocols are not designed to protect length and type of Record Layer, which underlies security sub-protocol in SSL/TLS. Wagner and Schneier analyze the SSL protocol and summarize significant attacks, which consist of traffic analysis attacks. They note that Bennet and Yee have explained how an examination of cipher text length can reveal information about URL requests in SSL-encrypted Web traffic [14]. Furthermore, more practical examples for actual web sites are described in the reports [7, 8]. They demonstrate that some web sites are distinguishable with a very high degree of accuracy using transfer data-size.

Hintz described a traffic analysis attack against SafeWeb [15], an encrypting web proxy. SafeWeb attempts to provide an anonymity service that hides the identification of its users from the web server and any monitoring system at their network connection. Hintz demonstrates that size of the transferred data can reveal the URL that the user accesses. Sun et al. also show that 100,000 Web pages [16] are identified by comparing captured traffic with the data sizes of these pages. Danezis enhances encrypted traffic analysis using the Hidden Markov Model [17]. Bissias et al. perform analysis with other encryption protocols such as SSH, IPsec and WEP/WPA, the conditions of which are harder than SSL/TLS. They analyze traffic without session reconstruction, which is effectively available in analysis of SSL/TLS.

However it is not feasible for encrypted traffic analysis to be applied to intrusion detection directly, because these attacks target confidential information encrypted by these protocols. These methods have to archive thousands of Web pages beforehand and compare captured traffic with these pages. To detect encrypted attacks, the process has to be as quick as possible and not depend on a relationship between actual web sites and captured traffic. The intrusion detection system only has to recognize the frequency of each access to perform frequency analysis. Our system analyzes traffic sequentially without pre-operation and does not reveal the relationship between encrypted and plain accesses, because the system distinguishes accesses by sequential temporary IDs.

6.3 Anomaly detection for web server

Some anomaly detections [18, 19, 20, 21, 22, 23] for web servers have been proposed. These methods extract parameters to identify accesses and then detect anomaly activities using some machine learning algorithms, for example SVM and HMM. In most cases, the approaches use length of request-line or a pair of attributes and parameters included in the request-line to discriminate accesses. Using encrypted traffic analysis, such specific parameters are not available, only the total size of requests and responses are available. Therefore, our conditions are much harder than those addressed by these methods. Vigna et al. [19] employ state transitions to track abnormal activities. The approach of state transition is also effective after encrypted accesses are identified by our access identification process.

7 Conclusion

In this paper, we proposed an intrusion detection system for encrypted web accesses. The proposed system detects attacks using transferred data size and timing, which are available without decryption. Though the approach is an application of a traffic analysis attack against security protocols, pre-operations are not needed and privacy is not violated. The detection is based on anomaly detection, which relies on the frequency of similar accesses and the characteristics of usual HTTP accesses, but the condition is much harder than in conventional anomaly detection. We evaluated the accuracy of the proposed system using an actual dataset gathered at a network gateway and a DARPA IDS evaluation dataset. We conclude that our system detects several kinds of attacks, even if the traffic is encrypted.

Acknowledgments
We would like to thank Dawn Song from Carnegie Mellon University for very helpful comments and suggestions.

References

[1] K. Hickman, "SSL 2.0 PROTOCOL SPECIFICATION", Available at: http://www.netscape.com/eng /security/SSL_2.html, 1995.

[2] A. Freier, P. Karlton and P. Kocher, "The SSL Protocol Version 3.0", Avaiable at: http://home.netscape.com /eng/ssl3/, 1996.

[3] T. Dierks and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, 1999.

[4] A. Hintz, "Fingerprinting websites using traffic analysis", Workshop on Privacy Enhancing Technologies, 2002.

[5] G. Bissias, M. Liberatore, D. Jensen, and B. Levine, "Privacy Vulnerabilities in Encrypted HTTP Streams", Workshop on Privacy Enhancing Technologies, 2005.

[6] Q. Sun, D. Simon, Y. Wang, W. Russell, V. Padmanabhan and L. Qiu, "Statistical identification of encrypted web browsing traffic", IEEE Symposium on Security and Privacy, 2002.

[7] H. Cheng And R. Avnur, "Traffic Analysis of SSL Encrypted Web Browsing", Available at: http://www.cs.berkeley.edu/˜daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps, 1998.

[8] S. Mistry and B. Raman, "Traffic Analysis of SSL-Encrypted Web Browsing", Available at: http://bmrc.berkeley.edu/people/shailen/Classes/ SecurityFall98/paper.ps, 1998.

[9] R. Lippmann, J. Haines, D. Fried, J. Korba, K. Das, "The 1999 DARPA off-line intrusion detection evaluation," Computer Networks 34, 2000.

[10] DARPA Intrusion Detection Evaluation. http://www.ll.mit. edu/SST/ideval/.

[11] Web Application Firewall Evaluation Criteria, http://www.webappsec.org/projects/wafec/, 2006.

[12] S. Goregaoker, "A Method for Detecting Intrusions on Encrypted Traffic", Technical Report TR-010703, Computer Science Dept., Florida State Univ., 2001.

[13] T. Leckie and A. Yasinsac, "Metadata for Anomaly-Based Security Protocol Attack Deduction", IEEE Transactions on Knowledge and Data Engineering, 2004.

[14] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol", 2nd USENIX Workshop on Electronic Commerce, 1996.

[15] SafeWeb, http://www.safeweb.com, 2002.

[16] DMOZ Open Directory Project, http://dmoz.org, 1998.

[17] G. Danezis, "Traffic Analysis of the HTTP Protocol over TLS", Available at: http://homes.esat.kuleuven.be/ gdanezis/TLSanon.pdf, 2002.

[18] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks", ACM Conference on Computer and communications security, 2003.

[19] G. Vigna, W. Robertson, V. Kher and R. Kemmerer, "A Stateful Intrusion Detection System for World-Wide Web Servers", Annual Computer Security Applications Conference, 2003.

[20] H. Kim, S. Cho, J. Seo, Y. Lee and S Cha, "Use of Support Vector Machine (SVM) in Detecting Anomalous Web Usage Patterns", Symposium on Information and Communications Technology, 2004.

[21] T. Konno and M. Tateoka, "Accuracy Improvement of Anomaly-Based Intrusion Detection System Using Taguchi Method", Symposium on Applications and the Internet Workshops, 2005.

[22] J. Estevez-Tapiador, P. Garcia-Teodoro and J. Diaz-Verdejo, "Detection of Web-Based Attacks through Markovian Protocol Parsing", IEEE Symposium on Computers and Communications, 2005.

[23] W. Robertson, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks", Network and Distributed System Security Symposium, 2006.