Mining Frequent Attack Sequence in Web Logs

Hui Sun, Jianhua Sun(✉), and Hao Chen

College of Computer Science and Electronic Engineering,
Hunan University, Changsha, China
{huisun,jhsun,haochen}@hnu.edu.cn

Abstract. As a crucial part of web servers, web logs record information about client requests. Logs contain not only the traversal sequences of malicious users but the operations of normal users. Taking advantage of web logs is important for learning the operation of websites. Furthermore, web logs are helpful when conducting postmortem security analysis. However, common methods of analyzing web logs typically focus on discovering preferred browsing paths or improving the structure of website, and thus can not be used directly in security analysis. In this paper, we propose an approach to mining frequent attack sequence based on PrefixSpan. We perform experiments on real data, and the evaluations show that our method is effective in identifying both the behavior of scanners and attack sequences in web logs.

Keywords: Log analysis · Web security · Web attacks · Sequential pattern mining

1 Introduction

The web is an important part of the Internet, and it becomes the largest publishing system in the world with its pragmatic natural attributes. With increased information sharing through network, attackers are attracted by the range and diversity of information, which causes the continued increase of attack frequency. They either endeavor to compromise the corporate network or the end-users access to the website by subjecting them to *drive-by-downloading* [13]. Among all these kinds of attacks, the attack for web servers is the most serious one with a variety of different ways, such as distributed denial of service and weak password guessing. Imaging that if a malicious user has intruded into a server that runs a database and network operating system, it would be relatively easy to obtain private information in the database or shutdown the network for a while. Under this circumstance, it is necessary to analyze the accident by finding attack footprints from log files, because the web logs contains the original information about client requests and run-time errors.

An entry will be added into access log when user request to the server, and the entry records client request information such as the time of request, client IP address, HTTP method, and so on. As one type of request users, attacker intrusions to server also be recorded. Thus, in web server's daily operation and

X. Huang et al. (Eds.): GPC 2016, LNCS 9663, pp. 243–260, 2016.
DOI: 10.1007/978-3-319-39077-2_16

security emergency response, web logs are always used to evaluate the risks of website. On the other hand, webmasters need to carry out investigation and evidence collection to find attackers by analyzing access log of the last few days, and then reconstruct the attack flow. A common method is using commands like "*grep*" (a tool using regular expressions to search text). In this way, certain keywords are searched in logs, then the records that contain specific keywords will be analyzed manually. However, this manual analysis is time-consuming and the analyzer is also expected to have a thorough knowledge on web security. Of course, there still exist many automatic analysis tools for web logs. For example, Piwik [3], Kibana [2], AWStats [1] and Splunk [4] are free open source software to analyze web logs. With these tools, we can perform searching, visualization, analyzing, and many other operations on web logs efficiently. The bad news is that these tools can just achieve some simple statistical data collection of deep analysis. Therefore, with data mining technology some researchers analyze user access patterns, which are called *Web Usage Mining*.

The data source of web usage mining mainly comes from access logs in web servers. Commonly, it is used to discover usage patterns and understand the need for web-based applications [5]. And web usage mining is widely used by companies [8,11,16] to improve service quality, to provide personalized services, and to find potential users. Actually, log mining in security plays a decisive role for webmasters and companies. It can be easier to know what the attacker is interested in, and whether 0-day is exploited in large-scale. However, web usage mining has been rarely applied to the field of web security. This paper focuses on the method of sequential pattern mining, which aims to learn the frequent attacking sequence from records logged by web servers.

In this paper, we make the following contributions:

- With sequential pattern mining, we can obtain frequent attacking sequences. In this paper, a method for mining frequent attacking sequences is proposed, which can not only reflect the attacker's intension, but also explore the common sequences of different security scanners.
- We distinguish scanners from malicious man-made attacks. The typical way of attacking a website is to first detect vulnerabilities of the website with scanners. Because the different behavior between the scanners and the manual requests, it is necessary to distinguish the man-made attack from the automatic tools before mining to make sure the accuracy of mined sequences.
- We explore the regular patterns of scanners. Scanners are typically black-box tools. It is beneficial to understand the internal working principle by mining attacking patterns of vulnerability scanners.
- We visualize the attack sequence with flowchart. Frequent sequence mining can help administrators to understand the attacker's behavior, which also can be utilized to display attacks sequence. The flowcharts are much intuitive for human investigation and can help webmasters to identify vulnerabilities to take further actions to protect the website.

2 Background and Motivation

This section presents an overview about the method of web usage mining and motivates the design of mining sequential attacking patterns. In addition, some representative web vulnerabilities that are often exploited in web applications are introduced in this part, which can help understand the typical payloads for each vulnerability.

2.1 Web Usage Mining

Generally, the data source of web usage mining comes from client click stream (always web server logs). The process is composed by three phases: the preprocessing, the pattern discovery, and the pattern analysis [12,14]. The preprocessing stage aims to clean up the raw data and transform the logs to what is more accurate and suitable for data mining. Transformed data are processed in the pattern discovery stage with specific data mining algorithms in order to find common rules. And in the final stage, useless patterns are filtered out.

Data Collection and Preprocessing. The raw documents are collected in this stage. The data for web usage mining usually come from three parts: server level, client level and proxy level. Among them, the server level is the most convenient for collection, including web log file, web page content and the website structure, etc. However, the collected data can not be used for modeling directly. It must be preprocessed to make sure the modeling in later steps more reliable. Ideally, the data should be in the format that just contains the page views of each user session. In general, the data preprocessing task involves data cleaning and filtering, user and session identification, and path completion [10,15,17].

- Clean up data. Filtering out error requests which are considered useless for understanding user actual behaviour. And then cleaning up the automatic requests (e.g., the requests for graphics files) which are not specifically requested by user. Another example for ineffectual requests in accessing records is web bots. Behaviour of web bots differs from human and not interested for web usage mining.
- User identification. The task of this part is to define each user, which help to implement access frequent sequence mining. The free-form structure of the Internet means that most users accessing on most websites are anonymous. The existence of local caches, corporate firewalls and proxies make it more difficult to identify each user. For instance, users who use the same proxy may access the site at the same time. Even more, the same user may request the web server on different machines or browsers. In order to improve the accuracy of user identification, lots algorithms have been proposed [6,18], while these algorithms have an adverse effect on realisation. Usually, web usage mining distinguishes each user by their IP address and operating system.
- Session identification. For each visitor, log files record which pages are requested, the order of the requests, and the duration of each page view.

Session identification divides each user's requests into many parts, which can reflect user continuous request to website [20]. A key factor for the quality of web usage mining is the real scale of session identification.
– Perform path completion. Not all pages requested by agent are recorded in log file. Just like when "back" button is pressed to return the page viewed before. The browser will go back to the page that has been cached locally rather than send a request to the web server again, and then the web logs will miss the record. This part tends to complete the paths which were requested by users but not recorded in the web logs [9].

Pattern Discovery Stage. The purpose of this stage is to establish the common patterns for users. The techniques in the pattern discovery stage include *statistical analysis*, *clustering and classification algorithms*, *frequent itemsets and association rules*, and *sequential patterns*. Among the above algorithms, statistical analysis, which is used to discover frequent access pages or average view time of a page, aims to improve the performance of the web system. At the same time, frequent itemsets and association rules also benefit to find out the frequent access pages or improve the struct design for the website. Clustering and classification algorithms divide the objects into multiple classes, and it is usually believed to be beneficial to market segmentation or information retrieval. Common sequential patterns expose users' visit patterns that reflect the access trends.

Pattern Analysis Stage. Not all patterns uncovered in the pattern discovery stage will be considered useful for analyzers. This stage filters out the rules and patterns that are common sense patterns. Hence, in this stage, we need to transform the patterns to comprehensible forms according to the actual application and keep the interesting, useful, and actionable patterns.

2.2 Web Application Security

According to numerous studies, the preferred method for attacking business's online assets is to exploit the vulnerabilities of their web applications. Vulnerability in web application is a security weakness that allows an attacker to bypass security checks or break system assurance. In the following, we discuss some common web vulnerabilities.

SQL Injection. In SQL injection, malicious codes are inserted into strings that are later passed to an instance of SQL Server for parsing and execution, which aims to get the higher authorized access in the database (e.g. administrator right to the database). For example, when users' input is not checked strictly, the request containing malicious codes will be submitted to the server, then the malicious statements are parsed into SQL statements to execute. SQL injection may lead to security risks such as the leakage of sensitive information, the modification and deletion of user data.

Cross-Site Scripting (XSS). XSS enables attackers to inject client-side script into web pages that will be viewed by other users. Most experts distinguish XSS

flaws into non-persistent and persistent. The further division of these two groups is reflected XSS, persistent XSS, and DOM-based cross-site. In reflected XSS, the bait is an innocent-looking uniform resource identifier (URL), pointing to a trusted site but containing a XSS vector. The injected script will be executed by the victim's browser when the URL is clicked. While in persistent XSS, the data provided by attackers are saved by the server, and permanently displayed on "normal" pages returned to other users in the course of regular browsing. The DOM-based cross-site neither need the parse of web server nor the response to the malicious code. It is triggered by the DOM parser of the client browser. A cross-site scripting vulnerability may be used to steal the session cookie and bypass access controls (such as the same-origin policy).

Directory Traversal. It exploits insufficient security validation of users' input file names that contain "/", so that malicious users can achieve directory jump to traverse to parent directory on the server. The goal of this attack is to grant an application the access to files that are not allowed by the default privilege. In general, the path "*/etc/passwd*" that contains the password is a common file in Unix to demonstrate directory traversal. In order to detect this vulnerability, users always restructure the URL that contains sub-strings like "*../*" or its escape character "*x5c./*".

File Inclusion. The occurrence of this vulnerability is due to that the developers insert reusable code into a file openly and include it when these functions are called. Because there is no strict filtering on the entry function when calling the public file, the client can submit malicious functions to make the server execute, and achieve evil purposes.

Command Execute. A special URL constructed by attackers, and the URL contains commands expected to be executed by the server. It exploits that the server doesn't check strictly for those functions to be executed, and achieve the goals of getting server information, executing command or some Java code of server system, and uploading file to the server.

2.3 Motivation

It is widely reported that more than half of the security breaches target web applications, which indicates the importance of strengthening the security level of web applications. Clearly, organizations need a way to replace fragmented and manual penetration testing. There is a great demand for automated tools, so they can protect their global application infrastructures.

Based on the background of web application security and web usage data mining, we propose to mine sequential patterns for the web logs from the perspective of managing website more securely. Concretely, we intend to obtain common behaviors of attackers or black-box scanners. Given that, we need to find the potentially malicious records from the logs by matching common attacking payloads, and then locate the vulnerabilities or reveal the connection between different vulnerable pages.

3 Design and Implementation

In this section, we present the design and implementation of our system, and the main components and data flow is shown in Fig. 1. Similar to web usage mining, firstly, we collect the raw data that are mainly extracted from access logs. In addition, we collect payloads for each vulnerability. Secondly, different from common web usage mining, we remove normal requests from the logs and retain the records that may contain abnormal behavior. Thirdly, we distinguish users and their sessions in logs. When we identify each user, we divide the users into two categories: black-box scanners and malicious users. Then, we transform the target data into sequence database. After that, we use the algorithm PrefixSpan to mine frequent attack sequence patterns from the sequence database. At last, visualizing attack sequence be realized by the *dot* language, before that we maximized the sequences generated in the mining stage.

3.1 Data Collection

As depicted in Fig. 1, this is the first task before we process the web data. In this section, for vulnerabilities we illustrate what payloads are, and then we present some real payloads for certain vulnerability.

Payload Collection. Our intention is to discover abnormal behaviors from recorded requests. Hence, before preprocessing, a reference should be indicated to tell the system what kind of log entries contain possible attacks. Essentially, besides the original web data, we also need to collect payloads for real exploits.

Definition 1. *The payload of each vulnerability is the most substantive characteristic string or statements contained in the request URL, with which we can determine that the request attempts to exploit the vulnerability.*

Some common web vulnerabilities have been demonstrated. For example, we can confirm that XSS exploit is attempted when the string "*<*" *<img src=javascript:*" appears in the URL. Usually, the payloads of the SQL injection

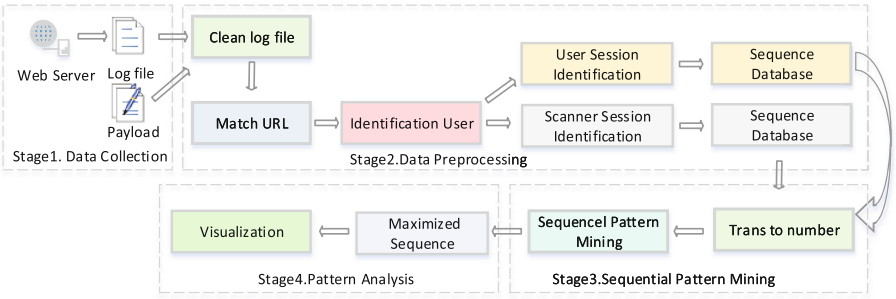

Fig. 1. The main process of frequent attack sequence mining.

```
DirectoryTraver:\..\..\..\..\..\..\                XSSAttack:>'><script>
DirectoryTraver:/\..\..\..\..\..\                  XSSAttack:>"><script>
DirectoryTraver:/????/????/????/????/????         XSSAttack:</textArea><script>
DirectoryTraver:/..%5c..%5c..%5c..%5c..%5c..       XSSAttack:%3cimg%20src%3
DirectoryTraver:/..%5c../                          XSSAttack:>"'><img src="javascript:alert
DirectoryTraver:/\x5c\x5c../                       XSSAttack:" style="background:url(javascript:
DirectoryTraver:..\../../..\                       XSSAttack:--><script>alert
```

Fig. 2. Part of payloads of XSS attack and Directory traversal.

attack contains basic database operations such as strings that contain "*select*", "*union*" in the URL. We gather many other payloads and corresponding attack vectors, and store them in a single file. Figure 2 shows some web application attack payloads that we collect.

3.2 Data Preprocessing

The results of preprocessing will be used for data mining algorithm directly. Therefore, the consequences, whether good or not, such as the accuracy of user identification and session identification, will straightly affect the mining outcome. This section mainly introduce the procedure of data clean-up, user and session identification method. It is worth mentioning, in the stage of user identification, we will divide users into malicious user and scanners in order to improve the accuracy of analysis.

Cleaning Up Log File. It is well known that if the type of web servers is different, then their log formats are also different. The most widely used free web server in Unix and Linux platform is *W3C*, *Nginx* and *Apache Server*, and *IIS* is the native web server in Windows platform. Factors like security, logs and statistics, virtual host, proxy server, and integrated applications, should be considered when choosing a web server. Each request from a client as well as the response from the server is recorded in the log file automatically. So an entry in the log file can reflect a particular action performed by the user. Meanwhile, the format of log file is often customizable, and each field is separated by spaces. A sample web log of *Nginx* is shown in Fig. 3.

- Remote host field: This field is located in the first parts commonly, and it records Internet IP address of the remote host.
- Author user field: It provides the authenticated user information. If this field is not a hyphen, it means that a client has gained access to directories which are password protected.

```
124.133.7.42 - - [04/Aug/2015:13:29:44 +0800] "GET /search?c=%3A%E4%B8%AD%E5%9B%BD&t=1
HTTP/1.1" 200 12189 "http://www.baidu.com/" "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.130 Safari/537.36" "-"
```

Fig. 3. A log record.

- Date/time field: This shows time zones and access time.
- Http request field: The Http request field consists of information that client's browser has requested to the web server. Essentially, this field may be partitioned into three areas: the request method, the uniform resource identifier, and the protocol.
- Status code: Obviously, not all client requests has succeeded, the status code field provides a three-digit response from the web server, indicating whether the request was successful or not, or if there was an error, we could know the error type. For example, 200 is OK, 304 is not-modified, etc.
- Transfer Volume (Bytes) field: This field expresses the size of the file sent by the web server to the client. Only when the GET request status code is 200, the number has a positive value.
- Referrer field: Listing the URL of the previously visited site that are linked to the current page.
- User agent field: This field is used to store client's browser and system information.

The log formats can be tailored to suit individual requirements. Certain fields might be chosen using the configuration file. We need to be aware of the structure of the log, and parse the string to IP, request time, method, URL, state and client field. The most common URLs are based on Http or Https protocol, and it is the breakthrough point of web security. Varieties of security threats are exploited by means of modifying the URL before sending requests to the server. It may cause a security problem if any level from the client to the server does not conduct filtering. According to this, we can find clues about attack by checking the record of the URL field.

After analyzing the structure of the log file, we can get the strings of attack payloads to match the URL, and then determine whether the user exhibit an attack behavior. Unlike common web usage mining whose data cleaning removes the exception requests and static requests, we remove normal requests, and retain the exceptional ones. More concretely, in this phase, if any payloads are contained in the URL, no matter which type of attack it belongs to, we store the record to a file suffixed with ".attack" and the string is reordered as access time, IP, browser and system of client, URL and status code. At the same time, we also maintain the type of each attack records. Figure 4 illustrates the segment of attack records that are generated after cleaning log files. Before user and session identification, the URL we got, is the raw request information of attacker, which might contain the string that is constructed by the attacker. And it is useless for analyzing the attack path, so we need to clean up the URL. We focus on the actual path that the attacker requests. In this step, we also transform the raw URL to the page it belongs to.

User Identification. Given that our goal is to get the user's common access sequence, we need to distinguish each user, and find the common attack behavior of them. The main task of user identification is to find out each attack who visits the website according to IP, system, browser and other information remained in the log after preprocessing. There are many effective algorithms proposed to

Attack Time	IP	Client Information	Vulnerability	URL	Response	Bytes
26/jun/2015 :15:20:50	61.185. 194.159	mozilla/5.0 (windows nt 6.1; wow64) gecko/20110613 firefox/6.0a2	DirectoryTraver	/util/ barcode.php?type=../../../../../../../../../../etc/ passwd%00	200	941391
26/jun/2015 :15:20:51	61.185. 194.159	mozilla/5.0 (windows nt 6.1; wow64) gecko/20110613 firefox/6.0a2	XSSAttack	/servlet/ %0arefresh:0;url=javascript:prompt(1)%0a1	200	941413
26/jun/2015 :15:20:51	113.247.2 22.22	mozilla/5.0 (windows nt 6.1; wow64; rv:6.0a2) gecko/ 20110613 firefox/ 6.0a2	DirectoryTraver	/sdk/../../../../../../../../../../../../etc/ passwd	400	941568
26/jun/2015 :15:20:52	113.247.2 22.22	mozilla/5.0 (windows nt 6.1; wow64; rv:6.0a2) gecko/ 20110613 firefox/ 6.0a2	Port DirectoryTraver	/?page=../../../../../../../../../../ etc/ passwd%00.jpg	200	941581
26/jun/2015 :15:21:01	113.247.2 22.22	mozilla/5.0 (windows nt 6.1; wow64; rv:6.0a2) gecko/ 20110613 firefox/ 6.0a2	XSSAttack	/?search =<script>alert(1)</script>	200	941572
26/jun/2015 :15:21:01	113.247.2 22.22	mozilla/5.0 (windows nt 6.1; wow64; rv:6.0a2) gecko/ 20110613 firefox/ 6.0a2	XSSAttack	/?search =%3cscript%3e alert(1) %3c%2fscript% 3e	200	941582

Fig. 4. The attack record of matching.

achieve user identification. The first general algorithm identifies different users by using the field of the IP and user agent. Others are based on cookies or extended property of the log. This method can distinguish users who are in the same proxy server effectively, and it provides higher accuracy. However, it requires both the server and the client to support cookie. In our implementation, we adopt the first method.

In daily operations of the website, besides suffering man-made attacks, the server is also scanned frequently by black-box testing tools. The man-made attack is different from automated tools. The records for scanners are not removed in the clean-up stage. The accuracy of the model mined from the data will be reduced if the two types of users mentioned above are mixed. Therefore, how to differentiate human attacks from scanners is also considered in order to improve the accuracy of analysis. The following are some methods that can be combined to make this distinction.

– Fingerprint of scanner. Different scanners usually have their own characteristics. A specific field such as the name of the scanner might be added into the request header.

- Trigger rules. Recording the times of users intercepted by the Web Application Firewall in a certain period. If the number is larger than the threshold allowed, you can assert the user is a scanner.
- Setting hidden links. The hidden links are invisible, and couldn't be clicked. While crawlers in scanners always catch all links including the hidden ones to detect vulnerabilities, so the hidden links are in the list of their requests. Especially, scanners based on Webkit will test the hidden links automatically to crawl more page to test. We can set a hidden link to induce scanners to request the link, which can help distinguish scanners from users.
- Cookie implantation. When the measured time under the condition of security rules triggered is lager than the threshold, a cookie is sent to the user. The client should carry the cookie when it requests the server next time, while most scanners can not achieve this operation. If the user requests next time without the cookie, we believe this user is a scanner. The advantage of cookie implantation is that it is more direct to find the scanner according to the next request.
- Response error ratio. This method is implemented by calculating the proportion of server response error in a certain duration, hence, it can detect sensitive directory scanning. Scanners based on dictionary file send request to each URL listed in the dictionary file, and then determine whether the path exists by getting the response returned by the server. By counting the number of return status of 404 for each user, we can ensure the user is a scanner when the number reaches a certain threshold.

Because the log data that we collect is in fixed format, considering the practical operability, we use the fingerprint approach to identifying scanners. In addition, we use the response error ratio method. We also consider if the number of attacks exceeds a certain frequency in a period. Combining these methods mentioned above with counting the number of response in the referrer field that are not 200, we can determine whether the user is a tool.

3.3 Attack Sequential Pattern Discovery

After preprocessing, two sequence databases are generated. One is the database of ordinary attackers, and the other is for scanners. We can conduct data mining algorithms on the two sequence databases separately to get the frequent sequence. In order to reduce the computational overhead, we convert each attack string to a corresponding number. Then, the pattern mining operation is performed based on these digital numbers.

Sequential pattern mining aims to find valuable patterns in a sequence database, which is based on a given mining support. More concretely, with mining we can find out all the frequent sequences for a given sequence database and minimum support threshold, and then remove the sequences that are duplications or contained by others. The *Sequence* is an ordered list of *itemsets* (A collection of one or more items, it can be expressed as $s = (x_1, x_2, x_3, \ldots, x_m)$, where x_k

represents an item), like $S = (s_1, s_2, s_3, \ldots, s_n)$, where s_k represents an itemset. The length of sequence, $|s|$, is the number of itemsets in the sequence. And the *Support* of a sequence W is defined as the fraction of sequences that contain W. *Frequent sequence* is a *subsequence* (A subsequence is that a sequence $< a_1, a_2, \ldots, a_n >$ is contained in another sequence $< b_1, b_2, \ldots, b_m > (m \geqslant n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$, such that $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$, …, $a_n \subseteq b_{i_n}$) whose support is $\geq minsup$.

Various sequential pattern mining algorithms have been proposed. Algorithms like AprioriAll [22], AprioriSome [21], GSP [22] are based on Apriori property to find the patterns in layers. The FreeSpan [7] and PrefixSpan are based on pattern-growth. The PrefixSpan is always preferable due to its performance and efficiency in large sequence databases, which is because that it generates less projection databases and less subsequence connections.

PrefixSpan. PrefixSpan is a kind of sequential pattern mining algorithm based on database projection, whose performance is better than GSP and AprioriAll. With the ability of handling large sequence databases, it is more widely used than other algorithms.

The *Prefix* in PrefixSpan means that: suppose that all the items are listed in alphabetical order. Given a sequence $\alpha =< e_1, e_2, \ldots, e_n >$ (where each e_i corresponds to a frequent element in S), and a sequence $\beta =< e'_1, e'_2, \ldots, e'_m >$ $(m \leqslant n)$ is called a prefix of α if and only if three conditions are satisfied: (1) $e'_i = e_i (i \leqslant m-1)$, (2) $e'_m \subseteq e_m$, and (3) all the frequent items belong to $(e_m - e'_m)$ are alphabetically after those in e_m. Conversely, *Suffix* is that given a sequence $\alpha =< e_1, e_2, \ldots, e_n >$ (where each e_i corresponds to a frequent element in S), let $\beta =< e'_1, e'_2, \ldots, e'_{m-1}, e'_m >$ $(m \leqslant n)$ be the prefix of α. Sequence $\gamma =< e''_m, e_{m+1}, \ldots, e_n >$ is called the suffix of α with regards to prefix β, denoted as $\gamma = \alpha/\beta$, where $e''_m = (e_m - e'_m)$[2].

PrefixSpan uses the divide-and-conquer strategy to generate more projected databases (let α be a sequential pattern in a sequence database S. The α-projected database, denoted as $S\ |_\alpha$, is the collection of suffixes of sequences in S with regards to prefix α), and the sizes of these databases are smaller than the raw sequence databases. The basic idea is to find out the frequent items whose frequency is greater than support, to generate their projection databases. For each projection database, the algorithm constructs the prefix pattern connected with the suffix mode to get frequent pattern.

Because Prefixspan does not need to generate candidate sequence patterns, it reduces the search space greatly. And it belongs to the growth pattern method, so compared with the original sequence database it reduces the size of projection database. The main cost of the algorithm is the construction of projection database. If the average length of sequences is large, it needs to build a projection database for each sequence pattern, then the time consumption increases correspondingly.

However, the average length of sequences in the database for scanners is too long for PrefixSpan. Certainly it will be, in scanning tools, tens of thousands of payloads waits to request each page of the website. In the course of our

Algorithm 1. Framework of our changed PrefixSpan.

Input:
The set of Sequence Database, S;
The minimum support threshold, $min\_support$;
Output:
The set of sequential patterns, S';

```
Scanning the Database S to extract the set of Items whose frequency is bigger
than min_support,
Items < −scan(sequenceDatabase)
for each item ∈ Items do
    α < −item;
    for all sequences ∈ sequenceDatabase
        SuffixSequence = Suffix(α).removeitem(α)
        S |α= AppendSuffixSequence(SuffixSequence);
    end for
    for all itemsequence ∈ α do;
        // extend the item in independence sequence like a.iadd() = {a, b}
        α′ < −item.iadd();
        l < −α′.length;
        prefixspan(α′, l, S |α′);
        // extension the item in a sequence like a.sadd() = {(a, b)}
        α″ < −item.sadd();
        l < −α″.length;
        prefixspan(α″, l, S |α″);
    end for
end for
```

experiment, we find that the lengths of sequences in scanning tools' database can reach hundreds. In the generated frequent sequence, we find that there are lots of repeated items, which means that the pages in the website is request back in attack path, like $< a(bg)aad >$, and we don't care about that. What we just want to know is whether the sequence contains more new paths, we just need the path like $< a(bg)d >$ rather $< a(bg)aad >$. So after generating the projection database, we remove all the items belonging to prefix sequence.

The algorithm is shown in Algorithm 1. Firstly, the sequence database is scanned to obtain all the frequent items N (the frequent sequence whose length is 1). Secondly, we divide the complete collection of frequent sequences into n subsets with different prefix. For each item whose frequency is bigger than $min\_support$, the corresponding projection database is obtained. Thirdly, all items that belong to prefix sequences are removed. Lastly, the main loop is executed until no frequent sequence is found.

We present an example to clarify our algorithm. Suppose that the sequence database is $[< a(ac)ad(cf) >,< (ad)c(bc)a >,< (ef)(ab)(df)ab >,< eg(af)cac >]$ and the $min\_support$ is 70 %. In the first step, we find the frequent items whose frequency is bigger than 70 %, and the results are a, c, d, f. Then, for each item, for example a, we get its suffix sequences from the origin database. For the first sequence, the

suffix for a is $< \_(ac)ad(cf) >$, and then we remove the item a from suffix sequence, leaving $< \_(\_c)\_d(cf) >$. Analogously, the new sequence database we get for a is $[< \_(\_c)\_d(cf) >, < (\_d)c(bc)\_ >, < (\_b)(df)\_b >, < (\_f)c\_c >]$. Next, we extend a to $< a, c >$ and $< (ac) >$, which aims to find new frequent sequence. If their frequency is bigger than *min_support*, like $< a, c >$, they are added to the final results as parts of frequent sequences.

3.4 Pattern Analysis

It is difficult to evaluate the vulnerability directly just with the mined result. First, lots of sequential patterns are generated after sequential pattern mining, and we need to analyze each pattern. Second, the format of sequences is represented with normal digital numbers, and these numbers are unreadable for the analyzers. For example, < 6 8 (12 8) 16 > is an example pattern, but it is meaningless characters if not transformed to a readable form. In order to ensure the accuracy and readability of the patterns, the following two steps are performed.

Step 1: maximizing frequent sequences. In the generated frequent sequences, a large number of sequences are redundant, because many sequences are contained in other sequences. At this point, we need to delete the redundant sequences. For example, for a frequent-sequence set $< a(bc) >, < a(fd)(bc) >, < a >$, we can find that the first and the third sequence are contained in the second one, so we need to remove the first and the third sequence, keeping the sequence $< a(fd)(bc) >$.

Step 2: transforming the sequential pattern into graphical representation. In order to make the patterns more intuitive to understand what the vulnerabilities are and where is vulnerable, we translate the resulting patterns into the *dot* language. Before that we parse numbers in frequent sequences to strings. In the step of sequential pattern discovery, we transform path strings to numbers that are stored in a file. In this part, we transform each number back to the corresponding string.

For example, Fig. 5 shows parts of frequent sequences. The sequences like < 0 >, < 0 2 >, and < 0 (2 6) 8 >, are contained in sequence < 0 (2 6) 8 (11 16) >, so in the first step, we delete these sequences. One of the final patterns in Fig. 5 is < 0 (2 6) 8 (11 16) 182 >. We translate the number

```
0 #
0 #2 #
0 #2 6 #
0 #2 6 #8 #
0 #2 6 #8 #11 #
0 #2 6 #8 #11 16 #
0 #2 6 #8 #11 16 #182 #
0 #2 6 #8 #11 16 #183 #
0 #2 6 #8 #11 16 #168 #
```

Fig. 5. Mined sequences.

```
fileinclude:/::0
xssattack:/::2
portsqlinject:/::6
clientsqlinject:/::8
clientfuzztesting:/::11
fromurlfileinclude:/::16
directorytraversal:/user::168
xssattack:/images::182
xssattack:/tpl::183
```

Fig. 6. Corresponding strings.

to strings according to Fig. 6, and the pattern is transformed to <*fileinclude:/ (xssattack:/ portsqlinjection:/) clientsqlinjection:/ (clientfuzztesting:/ fromurl-fileinclude:/) xssattack:/*>. Lastly, the sequence can be illustrated in *dot* language.

4 Experimentation and Analysis

In this section, we evaluate our implementation of frequent attack sequence mining. To prove the effectiveness of our system, we use the access logs from the real world. The experimental data are the raw access logs that are collected on a company's web server (Nginx) from September 2014 to October 2015. The testing machine is a 64-bit system with 4 GB RAM and a 3.2 GHz Intel i5 processor installed.

We first pre-process log files with different sizes separately. Figure 7 depicts the times consumed when processing different log files in our experiment. We can see that the time of finding attack record increases linearly, and this is because that the time spent on matching payload and reading log file is in proportion to the size of raw log files. While the sudden increase of time is because that attack file size increases, when writing the attack record to file it consumes a lot of time. On the other hand, from the red starts of Fig. 7, we can see that the size of attack files based on the number of attack record is randomly distributed. Intuitively, we may think that the probability of containing attack records for large logs is higher than small logs, but the appearance of attacks in the dataset is actually random. It implies that large-scale attacks may be less unpredictable.

With different sizes of attacker files that are generated by matching attack payloads, users and sessions are identified, and the results are shown in Fig. 8. For all benchmarks, the time increases with the increase of attack file sizes. For this part, we first use the method of fingerprint method, the response error ratio, and counting the number of attacks (which are introduced in Sect. 3.2). By these methods mentioned above, we separate man-made attack from scanner tools, and then, for each user, we identify sessions.

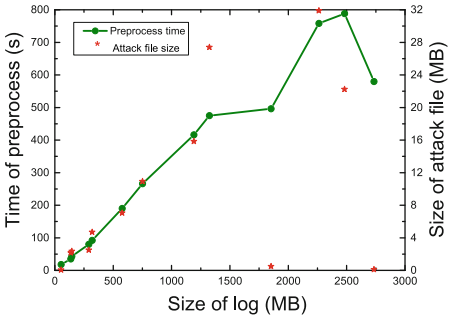

Fig. 7. Finding attack record time.

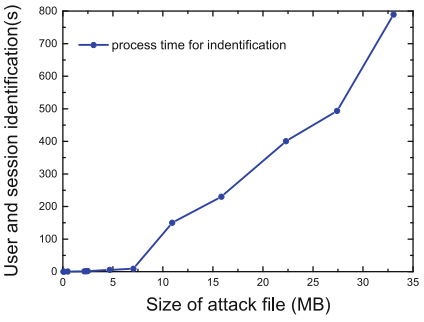

Fig. 8. User and Session identification time.

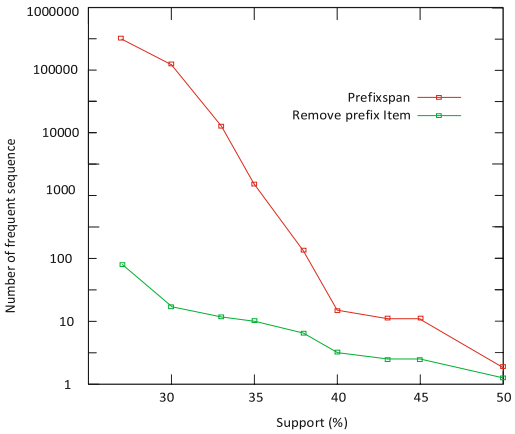

Fig. 9. Frequent sequences number of PrefixSpan and our algorithm.

After preprocessing log files, we have 52338 records of attack in total including 52090 records for scanners and 248 records for attackers. In order to evaluate the effectiveness of our improved Prefixspan algorithm, we compare it with the original Prefixspan. As shown in Fig. 9, with the decline of the support, the number of frequent sequences generated by PrefixSpan grow exponentially, while our algorithm generates much less frequent sequences than those that have the same support in Prefixspan. The main reason is that a large number of duplicate items has been removed after the projection database is generated. Correspondingly, the space of projection database is smaller.

In order to find sequential attack patterns from the man-made sequence database, we set the support with 3.5 %, 5 %, 7 %, 9 %, 10 %, 12 %, 14 %, 16 % to find more possible frequent attack sequences. Before generating the flowchart of attack, we transfer the sequence numbers to strings according to number-string file whose format is similar to Fig. 6. Then we transfer the patterns to the *dot* language. Table 1 shows the number of records generated with each support, and Fig. 10 illustrates the attack sequence generated by human when the support is 3.5 %.

Then, we experiment on the sequence database for scanners. As illustrated in Table 2, we set the supports to be bigger than that of the man-made, which is because that there are many common sequences between scanners. We do not need to understand the sequence of each scanner. What we really need to find

Table 1. Number of frequent sequence record by man-made

Support(%)	3.5	5	7	9	10	12	14	16	18
Number	27	14	8	6	5	5	4	3	3

Table 2. Number of frequent sequence record in scanning tools

Support(%)	22	25	27	30	33	35	38	40	43
Number	2936	132	89	35	17	11	8	5	4

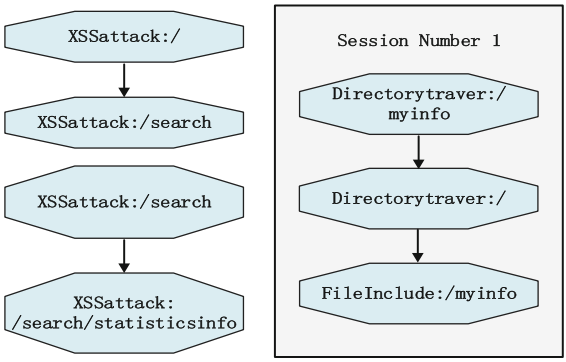

Fig. 10. The users' attack sequence of visualization.

is the common scanning sequence, and that is different from attackers' frequent sequences which are all important to understand the behaviors of attackers. So we set the supports with 22 %, 25 %, 27 %, 30 %, 33 %, 35 %, 38 %, 40 %, 43 %. Table 2 shows the number of records generated with each support, and we perform the same steps as for man-made attack. Figure 11 displays parts of the attack records of scanners when the support is 22 %.

By analyzing the results, we can conclude that for this website, the man-made attack sequence is biased to a specific page and a certain attack type,

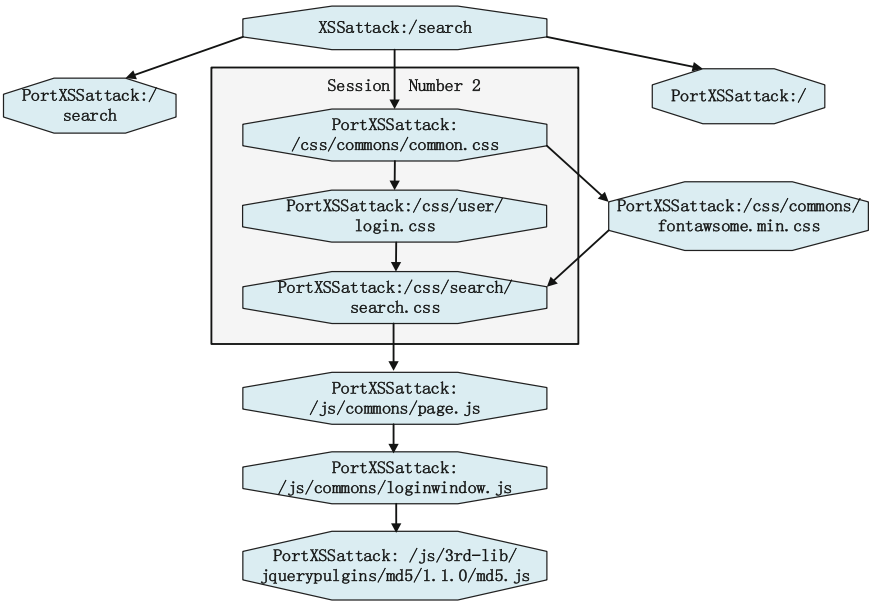

Fig. 11. The scanner attack sequence of visualization.

while scanners exhibit a wide varieties of attacks and more pages are involved. From the user attack sequence as shown in Fig. 10, we can see that the "/" page suffered from XSS attacks frequently, and the XSS attack against the page "*/search*" are also attempted. Or the attackers tried to use the XSS from page "*/search*" to "*/search/statisticsinfo*". Based on this, we can infer that there are contents that come from users' input, or the users are allowed to change the three pages that contain HTML tags. And XSS vulnerability may exist in these pages. In addition to the root directory, the attacker uses directory traversal to try to find the content in higher level directories to execute system command, and also perform "directorytraver" or "fileinclue" attack on page "*/myinfo*". Based on these information, the webmaster can check whether the pages of "/" and "*/myinfo*" can be bypassed by unprivileged users.

Figure 11 shows that the XSS attack sequence from scanners, it shows that for a certain type of general attack, the basic attacking paths share a lot similarity. Actually, many different types of attacks are found in the attack sequence, Fig. 11 only shows the XSS attack. With the patterns of attack sequences, we can distinguish scanners.

5 Conclusions and Future Work

This paper presents the design, implementation, and evaluation of sequential attack pattern mining in web logs. With efficient mechanisms, our system achieves the goals of discovering attack sequences of users and scanners to investigate the attack behaviors. We can pinpoint vulnerable pages and understand the internal-path of scanners. In the future, we plan to investigate new ways to identify the type of attacks to improve the accuracy, for example, using automatic classification algorithms. On the other hand, we are also interested in identifying black-box scanners by using the patterns generated by our system.

Acknowledgment. This research was supported in part by the National Science Foundation of China under grants 61173166, 61272190, and 61572179, the Program for New Century Excellent Talents in University, and the Fundamental Research Funds for the Central Universities of China.

References

1. Awstats. http://www.awstats.org/
2. Kibana. https://www.elastic.co/products/kibana
3. Piwik. http://piwik.org/
4. Splunk. http://www.splunk.com/
5. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. Knowl. Inf. Syst. **1**, 5–32 (1982)
6. Dziczkowski, G., Wegrzyn-Wolska, K., Bougueroua, L.: An opinion mining approach for web user identification and clients' behaviour analysis. In: 2013 Fifth International Conference on Computational Aspects of Social Networks (CASoN), pp. 79–84. IEEE (2013)

7. Han, J., Pei, J., Mortazavi-Asl, B., et al.: FreeSpan: frequent pattern-projected sequential pattern mining. In: Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 355–359. ACM (2000)
8. He, J.: Mining users potential interested in personalized information recommendation service. J. Mod. Inf. (2013)
9. Li, Y., Feng, B.Q., Mao, Q.: Research on path completion technique in web usage mining. In: International Symposium on Computer Science and Computational Technology, pp. 554–559. IEEE (2008)
10. Kewen, L.: Analysis of preprocessing methods for web usage data. In: 2012 International Conference on Measurement, Information and Control (MIC), pp. 383–386. IEEE (2012)
11. Mele, I.: Web usage mining for enhancing search-result delivery and helping users to find interesting web content. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 765–770. ACM (2013)
12. Nasraoui, O.: Web data mining: exploring hyperlinks, contents, and usage data. ACM SIGKDD Explor. Newsl. **10**, 23–25 (2009)
13. Provos, N., Mcnamee, D., Mavrommatis, P., et al.: The ghost in the browser analysis of web-based malware. In: Usenix Hotbots (2007)
14. Srivastava, J., Cooley, R., Deshpande, M., et al.: Web usage mining: discovery and applications of usage patterns from web data. ACM SIGKDD Explor. Newsl. **1**(2), 12–23 (2000)
15. Suresh, R.M., Padmajavalli, R.: An overview of data preprocessing in data and web usage mining. In: 2006 1st International Conference on Digital Information Management (2006)
16. Ting, I.H., Kimble, C., Kudenko, D.: Applying web usage mining techniques to discover potential browsing problems of users. In: IEEE International Conference on Advanced Learning Technologies, pp. 929–930. IEEE Computer Society (2007)
17. Varnagar, C.R., Madhak, N.N., Kodinariya, T.M., et al.: Web usage mining: a review on process, methods and techniques. In: International Conference on Information Communication and Embedded Systems (ICICES) 2013, pp. 40–46. IEEE (2013)
18. Wang, T., He, P.L.: User identification in web mining and iris recognition technology. Comput. Eng. **34**(6), 182–184 (2008)
19. Pei, J., Han, J., Mortazavi-Asl, B., et al.: Mining sequential patterns by pattern-growth: the PrefixSpan approach. IEEE Trans. Knowl. Data Eng. **16**(11), 1424–1440 (2004)
20. Qin, C., Liao, C.: Session identification based on linked referrers and web log indexing. Comput. Syst. Sci. Eng. **25**(8), 273–286 (2013)
21. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of ICDE, pp. 3–14. IEEE Computer Society (1995)
22. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996)