ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 545-554
doi: 10.18178/wcse.2018.06.093

The Research on Web Vulnerability Defense Model Based on SVM

Tao Feng [1+] and Tiantian Han [2]

School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730000, China

Abstract. Aiming at the problem of low defense efficiency of traditional web attack, this paper proposes a model of web attack defense framework based on SVM. It mainly focuses on cross-site scripting attacks and SQL injection attacks. And the defense against cross-site scripting attacks is not only for reflective and stored cross-site scripting attacks, but also for cross-site scripting attacks based on the DOM by modifying the scripting engine.

Keywords: web security, SQL injection attack, XSS attack, attack defense, SVM

1. Introduction

1.1. Research Background

With the rapid development of network and information technology, the security of information system is becoming more and more serious. Security incidents such as network attacks and intrusion frequently occur, and most of these incidents are caused by security vulnerabilities in the system. Of the various web security vulnerabilities, SQL and XSS have long been at the forefront of the list of vulnerabilities listed in the OWASP (Open Web Application Security Project) [1]. With the increasing complexity of web applications, there are more and more steps and attack points that may lead to SQL injection attacks and XSS attacks. Attack methods and deformation methods are also becoming more and more diversified, so the traditional defense method only relying on pattern matching based on precise feature rules is a bit difficult to deal with attack. Aiming at the above problems and referring to the research of machine learning algorithm in the field of intrusion detection, this paper proposes a model of web attack defense framework based on support vector machine, which is mainly aimed at the SQL injection attacks and XSS attacks with frequent and harmful effects. Among them, the traditional defense against cross-site scripting attacks can only be done on reflective and storage XSS attacks. In this paper, a scripting engine is introduced to improve the web attack defense framework to defense the cross-site scripting attacks based on DOM.

1.2. SQL Injection Attack

Each of web application uses the database as a background to store information. The background used by web applications is provided by the database system. SQL statements are created dynamically by the user-supplied data. These SQL statements directly interact with the database. SQL injection is a type of attack technique that injects SQL commands into user's input. Therefore, hackers try to inject illegal data into the input field. When malicious data is executed, unauthorized access occurs. The landing page is the ultimate goal of SQL injection attacks. For poorly designed website, insert the certificate directly into the SQL statement executed on the server side. An attacker can fake malicious SQL commands into the input field to avoid authentication [2-3]. The typical SQL injection attacks: anything'0R'x'='x.

1.3. Cross-Site Scripting Attack

[+] Corresponding author.
E-mail address: fengt@lut.cn.

Cross-Site Scripting (XSS) Vulnerabilities in today's Web application discovered vulnerabilities have become top-level vulnerabilities that are plague-like in modern web applications. XSS attacks are malicious attack vectors, especially in exponential growth. In addition, the results of XSS attacks can be seen globally through the entire World Wide Web. XSS attacks almost daily occur and some websites like Twitter, Facebook and Google have long since become victims of XSS attacks. XSS attacks allow attackers to run malicious script code on the victim's browser, leading to a wide range of side effects like data cracking, theft of cookies, passwords, credit card numbers, and more. As the most commonly used hacking technology, attackers use special symbols to convert data content to code content. Attackers may include scripts that can invoke the JavaScript interpreter. Cross-site scripting attacks are based on the principle that scripts can be hidden behind a URL or on a web page. In such an attack, the attacker can manipulate HTML tables, cookies, URLs, and external files. With this technology, hackers can execute illegal JavaScript code in other people's browsers. With the help of this technology, hackers are not targeting the victim directly, but rather exploiting the victim's vulnerability to accessing the site. Malicious JavaScript scripting is a legitimate part of the victim's user browser. Once the malicious script is successfully executed, hackers will be able to obtain the victim's confidential information, such as cookies, session id, etc. [4]. The typical XSS attack: <script type=’’text/javascrip’’>alert’’anything’’</script>.

Cross-site scripting (XSS) attacks are usually divided into three categories: Reflected XSS attacks, Stored XSS attacks, DOM-based cross-site scripting (DOM-XSS) attack. XSS cross-site scripting refers to malicious attackers using of web applications in the process of data input and output vulnerabilities, malicious code embedded in web pages or URL. When users browse the web or click on the link to access the site, malicious code is executed, and the user was to attack. In the traditional sense, XSS usually refers to reflective XSS and storage XSS. Attackers send malicious code, such as JavaScript code, to the server to execute locally malicious scripts by the client browser after the server returns the malicious data. However, the DOM (Document Object Model) -based XSS vulnerability is one of the major XSS vulnerabilities that specifically target web clients and does not rely on the client sending malicious data to the server executed locally on the client. Similar to reflective XSS, DOM-XSS usually entices users to access well-formed URLs to execute scripts, but the difference between the two XSS vulnerabilities is that the DOM-XSS URLs are not sent to the server, but executed locally in the user's browser. Thus the threatening is more and more difficult to defend.

1.4. Related Works

Traditional methods of detecting SQLI and XSS vulnerabilities and preventing attacks include code protection, static analysis, dynamic monitoring and testing. Each method has its own advantages, but all need to be strengthened. Code defenses are error-prone and require to use a security class library to rewrite the software; static analysis can generate error warnings but can’t generate concrete instances for detecting vulnerabilities; dynamic monitoring raises that runtime system overhead is too large; black-box approaches go through the web system to simulate real-world attack behaviour, inject malicious code into the web application, and analyze its reaction to detect the existence of SQL injection and XSS vulnerability. Due to the large variety of XSS input and SQL commands, as well as SQL injection and XSS malicious symbols for black-box detection, malicious statements have many manifestations, coupled with their respective encoding cloaked forms, which can be used for SOL injection attacks. The amount of malicious data is too large. In addition black box testing does not involve the internal components of the application. White box approach is mainly through the analysis of the source code of the web system. Testing and analysis may exist SQL injection vulnerability location, and the traditional white-box testing technology is not used to detect unknown vulnerabilities.

Xiang Fu and Kai Qian implemented SAFELI, a tool for detecting static vulnerabilities using symbolic execution. However, its detection depends on its attack model library and only applies to Web applications developed in the Java language [5]. The typical representative of SQL injection vulnerability detection defense using the combination of static analysis and dynamic analysis is a fully automated AMNESIA technology framework proposed by Halfond. The detection process is divided into two phases, that is, the static analysis phase and the dynamic analysis phase [6]. Sadalkar K et al. [7] proposed a model using a hybrid approach, namely static analysis and runtime analysis. This approach takes advantage of both static analysis and dynamic analysis, and is based on the MHAPSIA defense model proposed for the improvement of the model AMNESIA. The limitation of this approach is that it does not prevent certain deterministic SQL injection attacks, and some of the legal inputs that it builds into the query model are the same as the SQL injection inputs in some cases. And the query model is stored in the running array. There is no permanent storage source, and every application run need to build a security model query model. Literature [8] proposed a dynamic blob analysis method for DOM-XSS attacks. By constructing a DOM model and modifying the Firefox SpiderMonkey script parsing engine, a detection and defense against DOM-XSS attacks is implemented by using a dynamic stain analysis method. Putthacharoen R et al. [9] proposed a technique for changing cookies in some way, so that cookies are useless for XSS attacks. This technique is called "dynamic cookie replication" technology. The disadvantage of "dynamic cookie overwriting" technology is that protecting the user's cookie against XSS attacks, and is not exploited by the attacker, but it is a one-sided defense. Gupta S, Gupta B B [10] based on an extension of the above study propose a framework named XSS-SAFE that could be based on JavaScript disinfection conventional injection at source level and automatic feature injection. This server framework performs periodic rules, injection of signature content, and detection of components and HTTP response deviations for automatic disinfection of routine injections. They implemented their architecture model on Java and used functional parsers such as jsoup and Rhino for functional requirements. The downside is that although the false negative rate is zero, the false positive rate varies from 10% to 15%. Literature [11] respectively extracted the features of the collected URLs and JavaScript code in the normal page and XSS vulnerabilities page and set up the data set, and then used SVM, Naïve Bayes and J48 decision tree respectively to machine learning algorithms to learn. The experimental results show that the three machine learning algorithms are effective in detecting XSS attack, but the performance of SVM algorithm is better.

There are many major studies on hybrid defense SQL injection attacks and XSS attacks. Literature [12] proposed a technique for discovering SQL injection and XSS vulnerabilities in web applications. They implemented the technology in an automated tool called Ardilla. This proposed technique uses static code analysis to discover vulnerabilities. Sharma P et al. [13] proposed a hybrid query model generator. This model protects against SQL injection attacks by merging logic. This model incorporates algorithms for defense against reflective cross-site scripting attacks. The limitation of this approach is that its integration logic is used to defend against reflective cross-site scripting attacks only for certain deterministic types of reflective XSS attacks. And for more complex web applications it is less effective. A SECSIX security engine for preventing SQL injection attacks, reflective XSS attacks, storage XSS attacks and CSRF attacks was proposed in literature [14]. Saleh A Z M et al. [15] analyzed the shortcomings of scanners currently on the market and proposed solutions that could cover these drawbacks. The limitations of current vulnerability scanners are high false negatives. They used the boyer-moore string matching algorithm for vulnerability detection and performance analysis because of the lower processing overhead and high precision and no false positives. However, this approach is limited in web page level detection, and specific vulnerabilities in web applications are easily overlooked. In addition, testing for SQL injection with a large number of different point injections is not possible.

2. Proposed Solution

With the rapid development of web applications, the situation of web security is not optimistic. Although there are many detection and prevention methods for web attacks, with the diversification of attack modes and the ever-changing feature rules, blacklists or whitelists that are generated by established rules, and the traditional defense methods through static analysis and dynamic detection are more and more stretched. SVM has its good generalization ability and can find an optimal classification interval based on the principle of structural risk minimization. The examples are divided into two categories, which are widely used and concerned. SVM's self-learning ability can effectively improve the adaptability of web attack defense model. The solution proposed in this paper combines the traditional methods of defense, that is, static analysis and dynamic monitoring, and introduces the machine learning algorithm. By using SVM to improve the detection rate of the defense framework and reduce the false negative rate. The introduction of SVM into the framework of defensive web attack makes use of the self-learning ability and generalization ability of SVM to make up for the limitations of traditional defensive model in the face of diversified forms of attack. The frame diagram is shown in below Figure 1.

The solution is for web application security issues. This article focuses on the most common and most harmful SQL injection attacks and XSS attacks defensive study. The framework works mainly in two different modes, namely, security mode and real mode. In safe mode, it recognizes and marks hot spots. Each unique hotspot-id and tag number is assigned to each hot spot. Next is the plug-in module and feature extraction module. The main function of the plug-in module is called and filtered. The test code inserted in each hot spot is mainly used to perform security model analysis and real-time input analysis. The feature extraction module extracts the features of the web page data and the real-world input data in the secure mode invoked by the plug-in module, and then obtains the detection model by the training of the support vector machine through data pre-processing. In real mode, the real-time input is compared with the classification model learned and trained by support vector machine according to the characteristics of the collected pages in the process of safe mode. According to the comparison results, we can obtain the safety report.

In addition, DOM-XSS attacks are different from reflective and storage-based XSS attacks, that there are two characteristics: one is that the attack script does not appear in the original HTML text of the web page; the other is that the attack script does not need to be sent to the server. Confirming output on the client side by writing strict JavaScript scripts is an effective way to prevent DOM-based XSS attacks, but this approach often limits the flexibility of web applications. DOM-XSS attacks require some properties and methods that involve DOM objects. If through effective methods such as modifying the script parsing engine, the return page contains the above properties and methods of the JavaScript script to analyze, you can distinguish HTML text dynamically inserted by the web application developers and malicious HTML inserted by the user to the purpose of preventing DOM -XSS attack [16]. Due to the particularity of the DOM-XSS attack, this paper introduces a script parsing engine to find hidden pages according to the characteristics of its attacks, and obtains dynamically updated HTML texts from the DOM to detect and defend DOM-XSS attacks in real time. Whether a defense tool can detect DOM based XSS attacks depends on the HTML DOM object methods and properties implemented in the script parsing engine module. For complex scripts in web pages, all HTML DOM object methods and properties need to be implemented to detect if they are subject to DOM based XSS attacks.

2.1. Overall Architecture

Figure1 shows the overall architecture of the security engine in question. The proposed solution works in two different security modes, Safe Mode and Real Mode. The overall architecture is divided into four stages. The functional description of each stage is as follows:

Phase 1: The main tasks performed during this phase are scanning and identifying hotspots. For a given web application based on web developer convention that commonly used to express character characteristics of SQL injection, XSS attacks, and DOM-XSS related object methods and attributes as a standard, the built-in functions of the web application development language scan all documents, identify and mark hotspots. The hot spots are where the SQL query is executed and where the user typed it.

Phase 2: At this stage, plug-in test code is completed for the application's SQL injection attacks, reflected and storage XSS attacks, and DOM-XSS attacks. At the hot spot of each identification tag, add a snippet of code to invoke filtering. For SQL injection attacks, the code is inserted before the query is executed. Three types of XSS attacks are inserted after the user enters the code.

Stage 3: At this stage, the model is mainly generated, that is, the SVM learning training is used to generate the detection model. The built-in functions of the development language are used to parse the application programs. The program parsing module mainly analyzes SQL queries or user inputs and extracts the features according to the SQL query standards agreed upon by application developers and the commonly used signatures of XSS attacks. The detection model in safety mode is generated by using the normal features extracted from all the pages of web application on each hot spot and the automatic learning training using support vector machine through the collected web attack features.

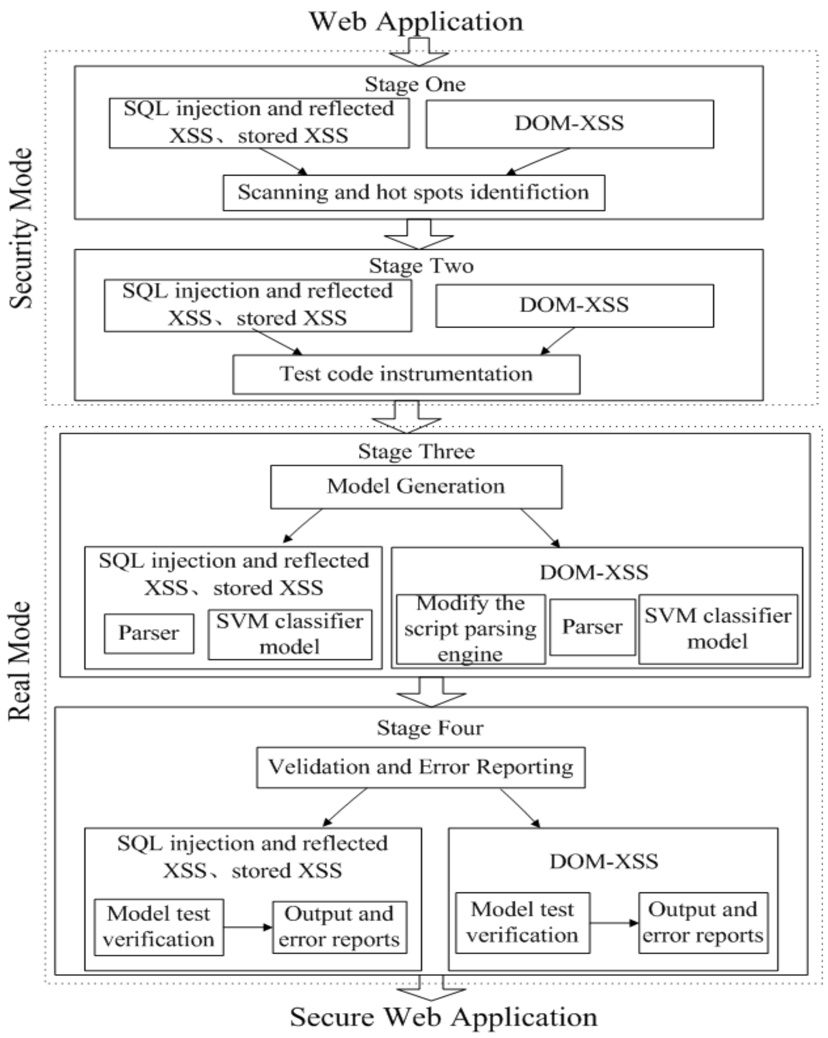
Web Application
Security Mode
Stage One
SQL injection and reflected XSS、stored XSS
DOM-XSS
Scanning and hot spots identifiction
Stage Two
SQL injection and reflected XSS、stored XSS
DOM-XSS
Test code instrumentation
Real Mode
Stage Three
Model Generation
SQL injection and reflected XSS、stored XSS
Parser
SVM classifier model
DOM-XSS
Modify the script parsing engine
Parser
SVM classifier model
Stage Four
Velidation and Error Reporting
SQL injection and reflected XSS、stored XSS
Model test verification
Output and error reports
DOM-XSS
Model test verification
Output and error reports
Secure Web Application

Fig. 1: The overall architecture of the web attack defense model.

In addition, due to the reason of the DOM-XSS attack is that a malicious script is inserted during dynamic DOM parsing of the static HTML text and displayed on the browser afterwards. Therefore, you need to change the DOM-XSS related object methods and properties by using a modified script parsing engine (such as Rhino) to find the hidden page before the program parsing module in real mode, so as to obtain the dynamically HTML text of the DOM updated.

In the model of SQL query and reflected XSS and storage XSS attacks detection, the detection model is divided into the detection model generated from the analysis of the security mode and the detection feature extracted from the real mode analysis. The detection model during security mode analysis contains all sensitive characters that are vulnerable to attack and are added to their blacklist along with their attributes. The detection features extracted during real mode analysis are compared with the detection models generated during the safety mode analysis. According to the test results, identify whether they have SQL injection and reflection or storage XSS attacks attempt.

In the detection model for DOM-XSS attacks, the detection model generated during security mode analysis contains sensitive characters in all DOM-XSS-related object methods and properties. The features extracted during the real mode analysis are compared with the detection models generated during the safety mode analysis. According to the test results, identify whether it DOM-XSS attack attempt.

Phase 4: At this stage, verification and error report are generated. The comparison between safety mode analysis and real mode analysis is displayed on the screen and an error report is generated. The returned error report contains the hot spot location and the detection result of the attack.

3. SVM

SVM (Support Vector Machine) was first proposed by Cortes and Vapnik in 1995. Due to its good generalization ability, SVM (Support Vector Machine) can find an optimal classification interval according to the principle of structural risk minimization. The examples are divided into two categories. The application of vector machine (SVM) technology has drawn extensive attention. The self-learning ability of In fact, the classification problem is to find a hyperplane that separates the two different categories well. There are many hyperplanes, and only one of the optimal hyperplanes is the one with largest distance from the two types of objects. Support vector machines are often used for prediction or classification [17]. SVM classification method is a small sample-based learning method that can map the feature vectors extracted and generated by network connection into a higher-dimensional space, and find a maximum interval hyperplane in this space that can achieve data classification. The greater maximum separation of data, the less data classification error is achieved. The classification results of SVM can be decided by a few support vectors. The complexity of the calculation depends on the number of support vectors, not the dimension of the sample, so as to avoid the dimensionality disaster. Support Vector Machine (SVM) classification algorithm is a kind of machine learning method. It needs to train the classification model by using the training set. And then we can use the classification model to predict the test set.

Support Vector Machine is a kind of machine learning method based on statistical theory. It seeks the best compromise between model complexity and learning ability based on limited sample information in order to obtain the best generalization ability. A support vector is a subset of a training dataset that is commonly used to define the boundaries of two types of data. Using a non-linear function, the sample data is mapped to a high-dimensional space. In the high-dimensional feature space, an optimal hyperplane is searched for the classification of the data set, which is determined by a certain number of support vectors.

For the sample$\{(x_1,y_1),(x_1,y_2),\ldots,(x_n,y_n)\}$, among them, $x_i \in R_n, y_i \in \{1,-1\}$， When y is 1, the sample belongs to the class of A. When y is -1, the sample belongs to the class of B. The classified boundary is expressed as：

$$\omega x + b = 0 \tag{1}$$

If a given sample set can be divided, there is a pair of $(\omega, b) \in R^n \times R$ to make：

$$\begin{cases} \omega x^i + b \geq 1, y^i = 1 \\ \omega x^i + b \leq 1, y^i = -1 \end{cases} \tag{2}$$

Its decision function is：

$$F_{\omega \cdot b}(x) = \mathrm{sgn}(wx + b) \tag{3}$$

Among them, ω is a weight vector, and b is a deviation value. The combined (2) is:

$$y_i(\omega x_i + b) \geq 1 \quad y = 1, -1 \tag{4}$$

At this point, the separation interval is $2 / \|\omega\|$， The maximum separation interval is equivalent to making the $\|\omega\|^2$ minimum. Therefore, the hyperplane that satisfies the (1) and the smallest $\|\omega\|^2 / 2$ is called the optimal classification surface. The optimal classification surface can be expressed as:

$$\min \Phi(w) = {}^{\|w\|^2}/_{2} \tag{5}$$

$$s.t. \;\; y_i(\omega x_i + b) \geq 1; i = 1,2, \ldots m \tag{6}$$

4. The Defense Model of Web Vulnerability Based on SVM

In the previous chapter, we mainly introduced the overall framework for web attack detection and defense, and the web attack detection and defense model based on SVM was introduced in this chapter. In this paper, through deep research and analysis of various forms of web attack variants, using support vector machines to automatically learn from the training samples to obtain rules to detect unknown data to determine suspicious attacks. The detection process mainly includes: feature selection and extraction, SVM training phase and classification phase. The following figure 2 is web attack detection model based on SVM, the main modules and steps described in detail in the following section.

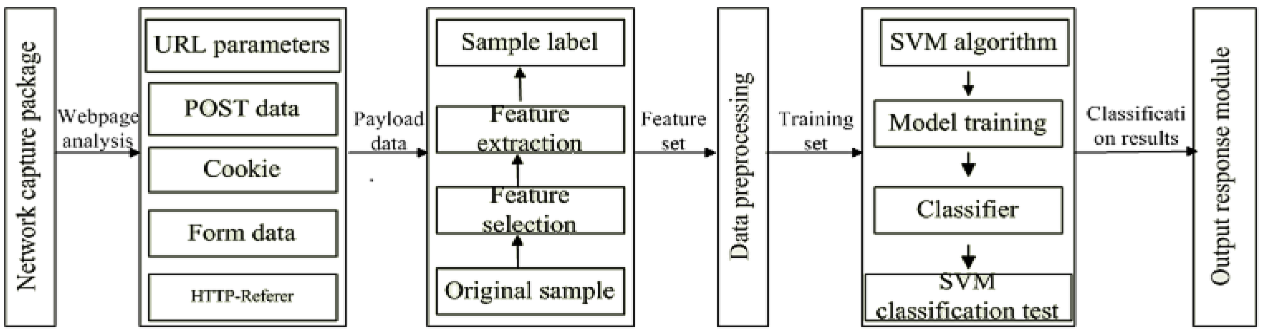

Fig. 2: Web attack detection model based SVM.

Feature selection is to select the subset of features that have the most important influence on the classification performance of the classifier from a group of features and to select the optimal feature combination without significantly affecting the classification detection rate. When detecting network attacks, a large amount of feature information needs to be extracted from the network data. However, with the complexity of attacks, the extracted feature dimensions also become very large. At the same time, the feature dimension extracted and processed is too high will lead to the decline of detection accuracy and real-time. Data feature extraction is a time-consuming and difficult task. Before data processing, removing the redundant or unimportant features of data and extracting only the key feature sets that effectively identify the behavior of the network will help improve the training speed and accuracy of SVM.

By deeply analyzing the SQL injection attacks and XSS attacks, analyzing the verification phase and utilization phase of vulnerability, various forms of SQL injection and XSS attack statements in various bypass scenarios, and various types of changes of SQL injection and XSS attack methods , based on the authors summarized and discovered the characteristics of the original payload data based on the two attacks, and we can summarize the following six features: Feature Key, Special Prefix Character Frequency, Various Special Characters such as Closure and Truncation Frequency (special characters Number of characters / number of characters), capital letter character frequency, number of digital characters frequency and space character frequency. Specific meaning as listed in Table 1.

Table 1: The name and meaning of the classified characteristics

Feature name	Characteristic meaning
Special keyword	Characteristic keywords for the corresponding type of attack
Special prefix character frequency	%, \, \ X, \ u, & #, & # x and other character frequencies
Special character frequency	Various closed, truncated and other special character frequency
Capital letter character frequency	Capital letters A-Z frequency
Number character frequency	Numbers 0-9 frequency
Space character frequency	The frequency of the space character

Among them, the SQL injection attack sentence common feature keywords are: and、Or、xor、substr、substring、len、length、mid、asc、having、back upload_file、load data infile etc..

XSS attack statements common feature keywords are: script、alert、prompt、location．Hash、src、href 、 @import 、 eval 、 XMLHttpRequest 、 ActiveXobject etc.. Common special characters are: “ !″# %&′ : ;〈 〉 = ?@[ ] \ / { } | $, ∗ + −.” etc..

The data preprocessing module converts the extracted combination of features into eigenvector data suitable for support vector machine processing. Because SVM can only deal with numerical data, we need to standardize and normalize the extracted data and complete the data type conversion.

SVM model training is mainly to train the input training set to produce data classification model. The training set is composed of feature vectors and their corresponding category labels, where the category label is a type of label that distinguishes between data types. In this paper, we use the "one to one" SVM multi-class classification algorithm. During the training phase, we construct two types of classifiers from two kinds of arbitrarily selected ones in this row, and form n(n-1)/2 SVM classifiers. In the classification stage, the n(n-1)/2 classifiers are respectively used to classify the input samples. Each time the sample is judged as a certain class, the type of the final decision is the final one for the sample data classification results.

5. Experiment and Result Analysis

In order to verify the effectiveness of the defensive framework proposed in this paper, DVWA is chosen as the testing environment for the security framework. The Tomcat server is installed on the Windows 7 operating system. MySQL and xampp are used to configure the DVWA test environment. Under Matlab R2011b implement the feature extraction and SVM classifier training, and test and simulation.

The entire testing process is divided into two modes, that is, safe mode and real mode. By analyzing the web server log in safe mode, we use the WebSpider to crawl the normal pages of DVWA, scan the crawled pages, mark the hot spots and plug the test code to get the normal sample set in the experimental data set. A hot spot is a sensitive place for attacks. For DOM-XSS attacks, in particular, the HTML text from the modified script parsing engine is scanned and hot spotted in conjunction with the DOM-prone properties and methods of the attack. In addition, from the vulnerability submitted site XSSED, exploit-db, HA.CKKERS and data sets HTTP DATASET CSIC 2010, we can collect the data of SQL injection and cross-site scripting attack sample set. In the experiment part, a total of 1200 data were collected as a training sample set of SVM, which included each 400 normal sample datasets, SQL injection attack samples and cross-site script attack samples. During the experiment, the kernel function of support vector machine was chosen as "liblinear", the penalty parameter C was set as 1, and the cross-validation method was used to analyze the training ability of the classifier. In this experiment, the triple cross-validation method was adopted. The dataset was equally divided into three equal parts. Two of them were used for training and the other one was used as the test set. The sample data used as the test set for the classifier was Unknown type of request data. The classifier generated by SVM training is the detection model in the framework of security defense. In the real mode, dynamic user input is scanned and identified by the same way; hot spots are marked; and then the test code used for calling is inserted into the test code. After the feature selection and extraction and data pre-processing, the detection model generated by the support vector machine is classified and tested to obtain the classification result and the corresponding data tag, which informed the type of attack. And from the hot id location mark, we can get the corresponding location of the attack occurred. Output the result report in the last stage.

In the test phase of DVWA, the selected sample set is to randomly select 300 pieces of data from the 1200 pieces of data for testing. Among them, the normal sample data, SQL injection attack and cross-site scripting attack sample data take each 100 pieces. In addition, in order to verify the ability of the security defense framework to deal with SQL injection and XSS attacks in other unknown ways, this article also collected 300 SQL injection and XSS attack sample data and 300 normal input samples in each attack database for security testing, and classifies the 900 attack data as unknown normal input, unknown SQL injection attack, and unknown XSS attack. Use the total of 1200 data to the web application DVWA which is under the application security defense for attack testing, and the results shown in below Figure 3. From the figure of the test results we can be seen that the detection rate is not 100%, the detection process there is a certain omission and false positives. Of these, 18 normal sample entries were falsely reported as attacks, 14 SQL injection attacks and 20 cross-site scripting attacks were undetected. 100 known normal samples, 100 known SQL injection samples and 100 known XSS attack samples extracted from the sample set of training support vector machines were completely detected and there was no omission. False positives and false negatives occur mainly in unknown samples and unknown attacks.

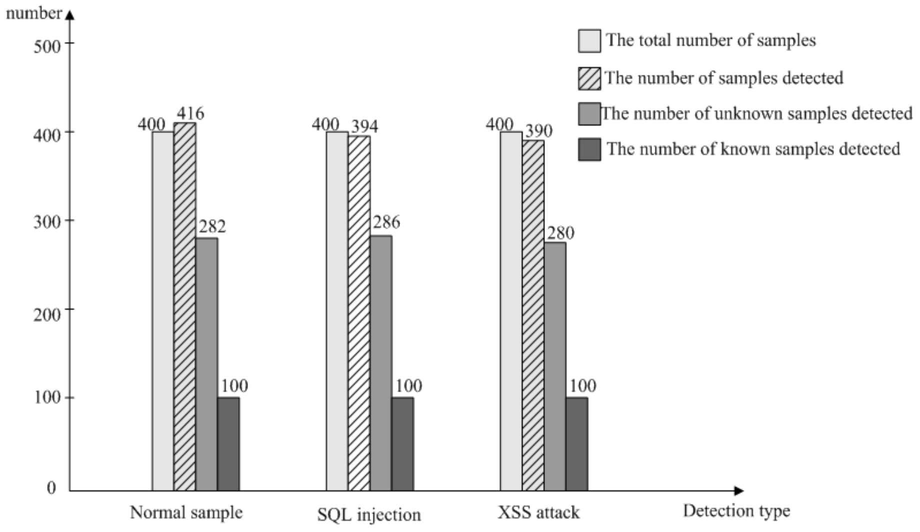

Fig. 3: The testing results.

The judgment of the detection performance of the proposed security defense framework is given by the following parameters: true positive rate (TPR), false positive rate (FPR), and false negative rate (FNR). Among them, the real rate is equivalent to the detection rate in a certain sense; the negative rate is equal to the false positive rate; and the false negative rate is equivalent to the false negative rate in a certain sense. Therefore, the security performance of the web attack security framework based on SVM in this paper is shown in Table 2. From the performance table, we can see that the proposed SVM-based web attack defense model proposed in this paper can effectively prevent SQL injection and three types of XSS attacks with a detection rate of 95% and above, with a false alarm rate of 5% the following. SVM-based security framework achieves a lower false positive rate than the higher false alarm rate of security frameworks such as XSS-SAFE [10] and SECSIX [14] using traditional defense methods. Although there is a certain omission rate and detection rate does not reach 100%, but this can be achieved by increasing the amount of training data support vector machine.

Table 2: Performance test results

category	TPR	FPR	FNR
Normal request	0.955	0.043	0.045
SQL injection attack	0.965	0.021	0.035
XSS attack	0.950	0.026	0.050

6. Summaries and Prospect

With the development of new technologies, web applications have exploded and new technologies will be used to the greatest extent. However, the safety awareness of web application developers has not been greatly improved with the development of new technologies. Therefore, the threats of various types of attacks on web applications will also increase. In particular, SQL injection attack and XSS attack. This paper thoroughly studied the two most widely attacked web attacks, namely, SQL injection attacks and three types of cross-site scripting attacks. The limitation of the traditional attack defense method is discussed. Based on this problem, a new defense model based on SVM is proposed, and the training model is learned by SVM. Compared with the traditional string matching and finite automaton, the query model is constructed. The method is more flexible, can achieve a lower false negative rate. With the use of various new technologies, new attacks and attack variants are becoming more and more numerous, so the defensive work on web attacks is a work that can’t be stopped from being researched. In the future work, we must further improve the detection of web attacks and defense accuracy and efficiency. In addition, based on the characteristics of

SVM, more training samples need to be collected to achieve a higher accuracy and lower detection rate of false positives and false negatives.

7. References

[1] Yu X Y, Chang Y, Xu H W, et al. Research and Implementation of Attack Defense Technology in Web Applications [J]. Computer Security, 2008 (6): 47-51.

[2] Halder R, Cortesi A. Obfuscation-based analysis of SQL injection attacks[J]. Proceedings - International Symposium on Computers and Communications, 2010:931-938.

[3] Zhang K X, Lin C J, Chen S J, et al. TransSQL: A Translation and Validation-Based Solution for SQL-injection Attacks[C]. International Conference on Robot. IEEE Computer Society, 2011:248-251.

[4] Hydara I, Sultan A B M, Zulzalil H, et al. Current state of research on cross-site scripting (XSS) – A systematic literature review[J]. Information & Software Technology, 2015, 58:170-186.

[5] Fu X, Qian K. SAFELI: SQL injection scanner using symbolic execution[C]// The Workshop on Testing, Analysis, and Verification of Web Services and Applications. ACM, 2008:34-39.

[6] Sadalkar K, Mohandas R, Pais A R. Model Based Hybrid Approach to Prevent SQL Injection Attacks in PHP[C]. International Conference. 2011:3-15.

[7] Halfond W G J, Orso A. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks[C]. Ieee/acm International Conference on Automated Software Engineering. 2005:174-183.

[8] Li J, Yu Y, Wu J S, et al. DOM XSS vulnerability detection algorithm based on dynamic stain analysis [J]. Journal of Computer Applications, 2016, 36 (5): 1246-1249.

[9] Putthacharoen R, Bunyatnoparat P. Protecting cookies from Cross Site Script attacks using Dynamic Cookies Rewriting technique[J]. 2011:1090-1094.

[10] Gupta S, Gupta B B. XSS-SAFE: A Server-Side Approach to Detect and Mitigate Cross-Site Scripting (XSS) Attacks in JavaScript Code[J]. Arabian Journal for Science and Engineering, 2016, 41(3):897-920.

[11] Vishnu B A, Jevitha K P. Prediction of cross-cite scripting attack using machine learning algorithms [C] //Proc of Int Conf on Interdisciplinary Advances in Applied Computing. New York: ACM, 2014: No. 55

[12] Kieyzun A, Guo P J, Jayaraman K, et al. Automatic creation of SQL Injection and cross-site scripting attacks[C]. International Conference on Software Engineering. IEEE, 2008:199-209.

[13] Sharma P, Johari R, Sarma S S. Integrated approach to prevent SQL injection attack and reflected cross site scripting attack[J]. International Journal of System Assurance Engineering and Management, 2012, 3(4):343-351.

[14] Nagpal B, Chauhan N, Singh N. SECSIX: security engine for CSRF, SQL injection and XSS attacks[J]. International Journal of System Assurance Engineering & Management, 2016:1-14.

[15] Saleh A Z M, Rozali N A, Buja A G, et al. A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm [J]. Procedia Computer Science, 2015, 72:112-121.

[16] Hang W F, et al. DOM XSS Attack Prevention Model [D]. Guangzhou University, 2010.

[17] ZHANG Yong-li, ZHU Yanwei. Application of improved support vector machines in intrusion detection[C]//Proceedings of the e-Business and Information System Security(EBISS). Tangshan, China: IEEE Express Conference, 2010: 1−4.