ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Security Engine for prevention of SQL Injection and CSS Attacks using Data Sanitization Technique

Kirti Randhe[1], Vishal Mogal[2]

Student, Department of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India[1]

Assistant Professor, Department of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India[2]

ABSTRACT: As dependence on web applications is increasing very rapidly in recent time for social communication, health problem, financial transaction and many other purposes. Web based systems consist of both infrastructure components and application specific code. The security motivation of web application developers and administrators should reflect the magnitude and relevance of assets they are suppose to protect. SQL injection attacks and cross site scripting attacks are the two most common attacks in web application. These attacks gives the attackers unrestricted access to the databases and even control of databases that underlay web application, which may contain sensitive or confidential information. This paper presents a new highly automated approach that underlay web applications against SQLI and XSS attacks. The most important advantage of this new mechanism is that it prevents all forms of SQL injection as well as Cross site scripting attacks. Presented approach in this paper is based on technique called reverse proxy. This technique is used to sanitize the user's input that may transform into database attack. Again a filter program is used which redirects the user input to the proxy server before it is sent to the application server. In the proxy server, data cleaning algorithm is triggered using a sanitizing application.

KEYWORDS: SQL attacks, SQL injection, Cross site scripting, Sanitization, Security Vulnerabilities.

I. INTRODUCTION

In this era where internet has captured the world, level of security provided by internet has not grown as fast as the internet application. Web services are also widely used to support many enterprise systems. Web applications are often vulnerable to attacks, which attackers intrude easily to the applications underlying database . SQL injection attack (SQLIA) is one of this techniques used to attack databases through the website. This attack tries to gain access to sensitive data directly by injecting malicious SQL codes through web application. SQLIA is the type of code injection attack in which an attacker uses specially crafted inputs to trick the database into executing attacker specified database commands. Cross site scripting is a type of code injection vulnerability that enables malicious user to send malicious script to web browsers. It occurs whenever a web application uses user inputs in response pages without properly validating them. When a user visits an exploited web page, the browser executes the malicious scripts sent by the application. This is called as XSS attack. Typical attack include installing malware, hijacking a users session or redirecting users to another site.

A. Basic Concepts

a.SQL Injection Phenomenon

In SQL injection attack, attacker inserts a malicious query into a web application by appending it to the input parameters.SQL injection vulnerability allows an attacker to flow commands directly to a web applications underlying database and destroy functionality or confidentiality.

b. SQL Injection Classification

- SQL Injection

SQL Injection Attacks (SQLIA) refers to a class of code-injection attacks in which data provided by user is included in the SQL query in such a way that part of the user's input is treated as SQL code. These types of vulnerabilities come among the most serious threats for web applications. SQL Injection vulnerabilities takes place because of nonexistent

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

and incomplete validation of user input. Due to this, an attacker can inject input that eventually alters the behaviour of the script being executed [6, 8]. SQL Injection Attacks can be carried out in various ways like using UNION keyword, Tautology condition, Group by Having Clause etc. There are different ways available for performing such attacks which are discussed in [4], [5], and [7] by different authors.

- Tautologies

The main aim of tautology-based attack is to inject code in conditional statements so that they are always evaluated as true. Using tautologies, the attacker can either bypass user authentication or insert inject-able parameters or extract data from the database. A typical SQL tautology has the form, where the comparison expression uses one or more relational operators to compare operands and generate an always true condition. Bypassing authentication page and fetching data is the most common example of this kind of attack. In this type of injection, the attacker exploits an inject-able field contained in the WHERE clause of query. He transforms this conditional query into a tautology which causes all the rows in the database table targeted by the query to be returned

- Logically incorrect query attacks

The main goal of the Illegal/Logically Incorrect Queries based on SQL Attacks is to collect the information about the back end database of the Web Application. When a query is rejected, an error message is returned from the database which includes useful debugging information. This error messages are helpful to the attacker to find vulnerable parameters in the application as well as database of the application. In fact attacker injects large input or SQL tokens in query to produce syntax error, type mismatches, or logical error by purpose. In this example attacker makes a type mismatch error by injecting the following text into the input field:

1) Original URL: http://www.goodsmarket-al.com/veglat/?id_nav=2234
2) SQL Injection: http://www.goodsmarket-al.com/veglat/?id_nav=2234'
3) Error message showed: SELECT name FROM Employee WHERE id=2234\'. From the message error we can find out name of table and fields: name; Employee; id. As the attacker gets updated information time to time, he can organize more strict attacks.

- Piggy-Backed Queries

The main aim of the Piggy- Backed Query is to execute remote commands or add or modify data. In this a type of attack, an additional query is injected by attacker along with the original query, which are said to "piggy-back" onto the original query. As a result, the database receives multiple SQL queries for execution. Vulnerability of this kind of attack is dependent of the kind of database [5].

- Inference

The main aim of the inference is to change the behaviour of a database or application. There are two well-known attack techniques that are based on inference: blind injection and timing attacks.

- Blind Injection

Eventually developers hide the error details which is advantageous for attackers to compromise the database. In this condition attacker face to a generic page provided by developer, instead of an error message. So the SQLIA would be more difficult but not impossible. An attacker can still steal data by asking a series of True/False questions through SQL statements.

- Union Query

The main aim of the Union Query is to trick the database to return the results from a table different to the one intended. By this technique, attackers join injected query to the benign query by the word UNION and then can get data about other tables from the application. This technique is mainly used to bypass authentication and extract data

- Cross Site Scripting

Cross Site Scripting is another common web application attack technique. In this technique code injection takes place that enables malicious user to send malicious scripts to web browsers. XSS allows attackers to execute script in the victims browser, which can create various security violations such as account hijacking, data theft, cookie theft, deface websites and denial of service. There are three types of XSS attacks: Stored, Reflected and DOM based. Stored attacks are also called as Persistent attack, these attacks occurs when the malicious code is submitted to a web application where it is stored permanently. These attacks generally occurs in forums, blogs or in social networking sites.
Reflected attacks are also called as non-persistent attacks, these attacks takes place when the server does not properly sanitize the output server to a visiting web browser /client.

II. RELATED WORK

SQLrand [3] appends a random token to SQL keywords and operators in the application code. This approach was created randomize instances of the SQL query language, by randomizing the template query inside the CGI script and the database parser. The drawback of this approach is that the secret token could be guessed every time, which makes the approach ineffective, and so this approach requires the deployment of a special proxy server.

In SQL Guard and SQL Check [8,9] queries are checked at runtime based on a model which is expressed as a grammar that only accepts legal queries. SQLCheck identifies SQLIAs by using an augmented grammar and distinguishing untrusted inputs from the rest of the strings by means of a marking mechanism. SQL Guard examines the structure of the query before and after the addition of user-input based on the model. In this two approaches developer have to modify code every time to get a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generated query.

CANDID[5] modifies web applications written in Java through a program transformation. This tool dynamically searches the programmer-intended query structure on any input and detects attacks by comparing it against the structure of the actual query issued. CANDID's simple approach turns out to be very effective for detection of SQL injection attacks.

AMNESIA[7] states combination of static analysis and runtime monitoring. In static phase, building models of the different types of queries consisting of application which can legally generate at each point of access to the database. In dynamic phase queries are intercepted before they are sent to the database and are checked against the statically built models. Queries that violate the model are prevented from accessing to the database. The primary drawback of this tool is that its success is dependent on the accuracy of its static analysis for building query models.

SQLIDS [11] comprises security specifications that describe the intended syntactic structure of SQL statements that are generated by the application. The detection technique is based on the criteria on which injected SQL commands have differences in their structure with regard to the expected SQL commands that are defined in the scripts of the web application. Therefore, if the required structure of the expected SQL commands has been previously determined, it is possible to detect malicious changes that modify this structure.

An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service identifies the SQLIA by using web services oriented XPATH authentication Technique [2]. Detection and prevention of SQLIA is done with runtime monitoring, in this the operation login page is redirected to their checking page.. User input field is compared with data existed in XPATH validator if it is identical then authorized user is allowed for next processing. Database in database server is not allowed to access directly by Web Service Oriented XPATH Authentication Technique .

SQLProb [6] constitutes the extracted user input data in the context of the syntactic structure of the query can be evaluated. This approach is fully modular and does not require access to the source code of the web applications or the database. Making use of this approach the system is easily deployable to existing enterprise environments and can protect various front-end web applications without any modifications.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

III. PROPOSED FRAMEWORK FOR DATA SANITIZATION

The block diagram of Reverse Proxy Model is illustrated in Fig. 1. Reverse proxy server is placed in between the client (request) and the server (response). The client is not aware of the presence of the reverse proxy server.

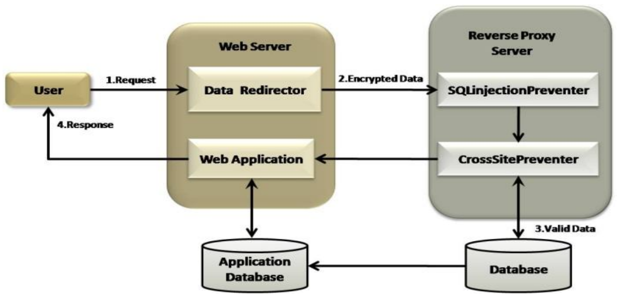


Fig. 1 Block Diagram of Reverse Proxy

Reverse proxy server is added in sanitization application for SQLIA and XSS attacks. This reverse proxy technique is used to sanitize the request from the user or client. The number of reverse proxies can be increased to handle the increased number of requests coming from the client, which enables the system to maintain the least possible response time even at very high loads. Reverse proxy server contains intrusion detection and prevention mechanism . The basic functional requirements of the web intrusion prevention technique are to receive client requests and server responses, which analyse the requests and response messages, decrypt HTTPS messages, forward the safe messages and reject the dangerous ones, and provide long lasting client requests and detective results. The proxy with intrusion prevention is divided into different parts. First is the request listener, it listens to the ports of reverse proxy server in real time and receives client requests and server responses. Next is the data redirector which extracts the client requests and create tokens. All the tokenized parts are saved in a hash table. The tokenized query is then compared with the existing document. Then the query tokens are transformed into XML format. The attacks are matched with the URL parameters of the HTTP header and the variable pattern is expressed by regular expression.

The SQL injection preventer and Cross site preventer analyse the client IP addresses. If attacks are encountered more than three times from a particular IP, the IP address is blocked ;otherwise the request will be forwarded to the next module.. In order to cheat and cross the existing IDS, many hackers use various methods allowed by the web server to encode part of HTTP header or payload, such as URL encoding. Through the encode analysis, it’s easy to identify the content of SQL injection and XSS attacks. SQL injection preventer and Cross Site Scripting preventer modules are used to check the intrusion, and if attack action is found, client request is denied.

The use of log module is to log the client access requests, discovered attacks, identify privacy leaks and other system actions. Logs can help the administrators to analyze system performance, track and position of the invaders.

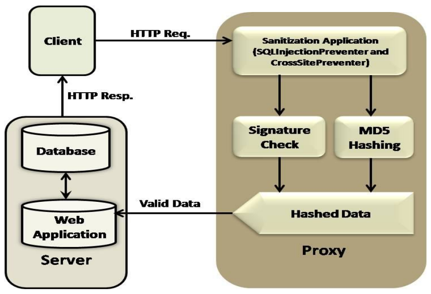


Fig. 2 Block Diagram of Sanitization Framework

Fig. 2 illustrates the block diagram of sanitization application. HTTP request is forwarded to the proxy, which authenticates the request by sending it to the sanitization Application. The sanitization application consist of SQL preventer and Cross site preventer which is divided into two parts i.e. signature check and MD5 hashing. The sanitization application extracts the URL from the http request and separates the URL and user data. Then the URL and user data is parsed into token [12]. First, Signature check validates the URL by using regular expression and if any invalid character is found in the input query it is marked as malicious and not forwarded to the next module. Only request which are reported as valid by signature check are forwarded to the next module. User data is parsed into tokens. These tokens are checked with the reserve SQL keywords and converted to corresponding MD5.The sanitization application filters the request for invalid tags and encodes it before forwarding to the server. Functionality of signature check and MD5 hashing modules is independent in a sense that the valid request goes to both the modules before getting executed on the web server.

Input: User data and extracted URL from the HTTP

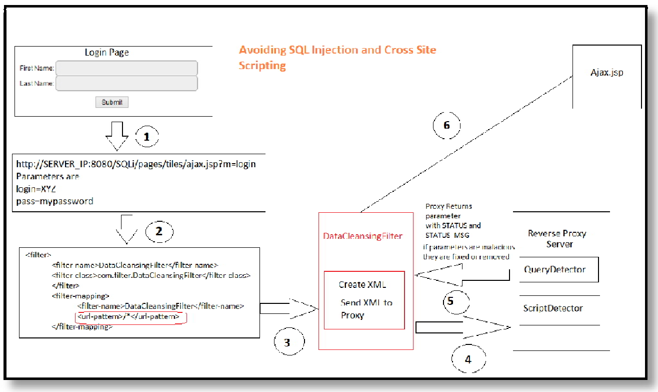
Output: Access to valid account

- SQL Injection preventer

1. Take the user request.
2. Parse the user request and store in the hash table and set a key value.
3. The parse or tokenize request checks using signature check through the regular expression.
4. If the tokenize request matches with AND_OR_Integer pattern discard the request. (AND_OR_Integer pattern checks the integer.)
5. If the tokenize request matches with AND_OR_String pattern discard the request. (AND_OR_String pattern checks the string & and/or within single or double quotes.)
6. If the tokenize request matches with SQL Statement Pattern discard the request. (SQL Statement pattern checks the SQL keyword.)
7. Else call the installed Cross Site Scripting preventer program for checking XSS attack.

- Cross Site Scripting preventer

1. Take user request in the form of HTML text.
2. Parse the request and store in the hash table.
3. Check the parse request using signature through regular expression.
4. If the parse request matches with the forbidden tag then discard it.
5. Else remove the tags and encode the URL.
6. If the parse request matches with the attribute tag like href, src and onclick remove that tag and encode the URL and push the tag in the stack.
7. Else remove unknown tags.


Login Page
First Name:
Last Name:
Submit
Avoiding SQL Injection and Cross Site Scripting
Ajax.jsp
1
2
3
4
5
6
DataCleansingFilter
Create XML
Send XML to Proxy
Proxy Returns parameter with STATUS and STATUS_MSG
if parameters are malacious they are fixed or removed
Reverse Proxy Server
QueryDetector
ScriptDetector

```
http://SERVER_IP:8080/SQLi/pages/tiles/ajax.jsp?m=login
Parameters are
login=XYZ
pass=mypassword
```

```
<filter>
        <filter name>DataCleansingFilter</filter name>
        <filter class>com.filter.DataCleansingFilter</filter class>
        </filter>
        <filter-mapping>
                <filter-name>DataCleansingFilter</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>
```

Fig. 3 Avoiding SQL Injection and CSS Attacks

IV. MATHEMATICAL MODEL

1) S= Σ (I, O, f(x))) where I is set of inputs , O is set of output, f(x) is set of functions ,s is system

2) I= { Q1, Q2…….Qn| S1,S2,…….Sn} where Q1,Q2…Qn are set of queries Q and S1,S2,….Sn are set of scripts S

3) O = v alidity {O1, O2}O1 is valid query or script

O2 is invalid query and script
4) f(x) ={ QD,SD,CV} QD is query detector
SD is script detector CV is validity
5) Q{ Type of injection attack Q1, Q2,……..Q5} where Q1 is Tautology, Q2 is Illegal/Logically Incorrect Queries, Q3 is UNION Query ,Q4 is Stored Procedures , Q5 is Piggy-Backed Queries
6) S{ S1,S2, S3 script attributes} where s1 is href, s2 is style , S3 is embedded script
7) P1={P1,P2,….Pn} where p is the set of all expect patterns of queries and scripts.
8) If P= Q|S Then attack alert and
9) If p $\neq$ Q|S then valid user.

V. SIGNATURE CHECK

The signature check handles all possible methods of SQL injection attack through regular expression. The URL is extracted from the HTTP request and the URL is tokenized. The regular expression checks the small parts of URL. If the regular expression identifies any form of SQL injection signature then the request is marked as malicious else it is marked as benign.

VI. OPERATING ENVIRONMENT

Operating System: Windows 7
Tool Used: Eclipse 3.7,JAVA
Database: MySQL
Database Connectivity: JDBC
Server: Apache Tomcat Server
Front End: HTML,Javascript,Cascading Style.

VII. EVALUATION AND EXPERIMENTAL RESULTS

The system implements new prototype version of reverse proxy server which prevents the SQL Injection Attacks(SQLIA) and Cross Site Scripting(CSS) Attacks. For the evaluation experiment, we used Apache Tomcat server, Mysql and JAVA web applications. The server runs on Windows 7 and the web server is Apache Tomcat. This server hosts a JAVA web application that uses Mysql database.To demonstrate the proposed system we have implemented following three modules.

- SQL Injection Preventer
- Cross Site Scripting Preventer
- Analysis Module

The user sends the input through the login page of web application. The data filter program, which is installed on the server gateway, gets the user input and redirects the request to the reverse proxy server. The reverse proxy server has the SQL injection preventer module and cross site scripting preventer module programs installed in it. The SQL injection preventer module validates the request against SQLIA and tokenises the request. Cross site scripting preventer module validates the request against XSS attacks by carrying out signature checks through the regular expression.

The analysis module checks attacker's activities. If the attacker attacks more than three times consecutively the IP address of the attacker will be blocked .

A. Runtime Analysis

In fig.4 we observe the execution time between the existing technique and proposed technique. One axis contains execution time in millisecond and other one contains total number of user request. From this analysis we can confirmly say that proposed technique required less execution time as compared to existing technique.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

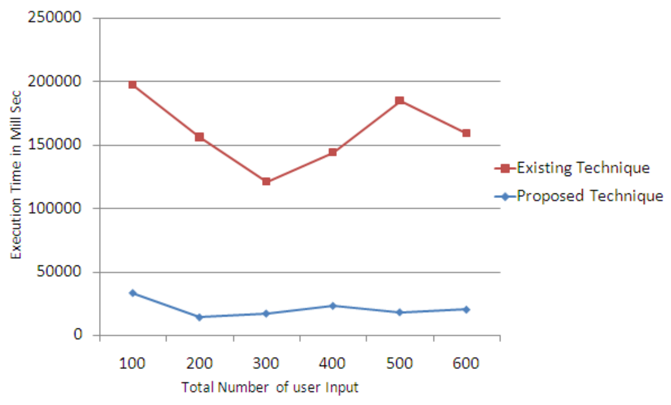


Fig. 4 Graph of Runtime Analysis

B.Output and Snapshots

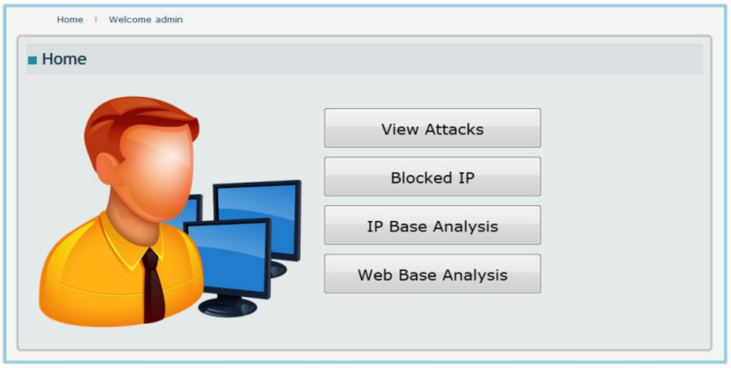
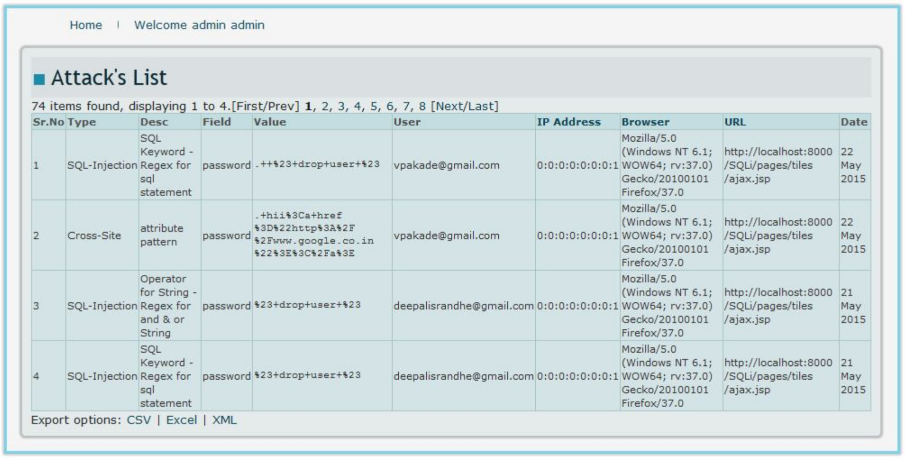


Fig. 5. Administrator Console of Analysis Module

"Administrator Console of Analysis Module": The administrator can review occurrences of attacks, block, IP addresses and other actions by using admin console.



Home | Welcome admin admin

Attack's List

74 items found, displaying 1 to 4.[First/Prev] **1**, 2, 3, 4, 5, 6, 7, 8 [Next/Last]

Sr.No	Type	Desc	Field	Value	User	IP Address	Browser	URL	Date
1	SQL-Injection	SQL Keyword - Regex for sql statement	password	.++%23+drop+user+%23	vpakade@gmail.com	0:0:0:0:0:0:0:1	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0	http://localhost:8000 /SQLi/pages/tiles /ajax.jsp	22 May 2015
2	Cross-Site	attribute pattern	password	.+hii%3Ca+href %3D%22http%3A%2F %2Fwww.google.co.in %22%3E%3C%2Fa%3E	vpakade@gmail.com	0:0:0:0:0:0:0:1	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0	http://localhost:8000 /SQLi/pages/tiles /ajax.jsp	22 May 2015
3	SQL-Injection	Operator for String - Regex for and & or String	password	%23+drop+user+%23	deepalisrandhe@gmail.com	0:0:0:0:0:0:0:1	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0	http://localhost:8000 /SQLi/pages/tiles /ajax.jsp	21 May 2015
4	SQL-Injection	SQL Keyword - Regex for sql statement	password	%23+drop+user+%23	deepalisrandhe@gmail.com	0:0:0:0:0:0:0:1	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0	http://localhost:8000 /SQLi/pages/tiles /ajax.jsp	21 May 2015

Export options: CSV | Excel | XML

Fig.6. Attack List of Analysis Module

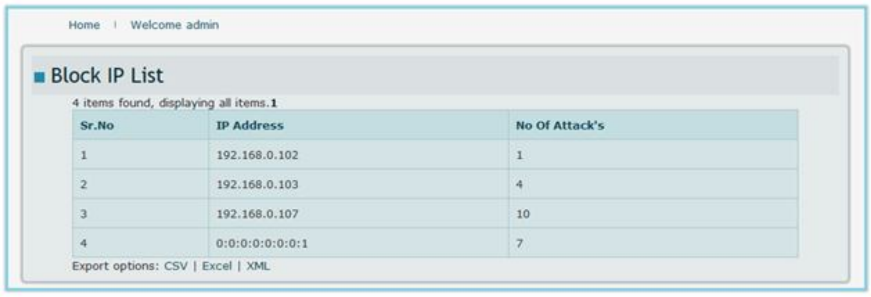
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

"Attack List of Analysis Module": The system provides a browser-based interface for the server administrator to monitor the system for possible attacks. The administrator can review occurrences of attacks, block IP addresses and other actions on the UI.


Home | Welcome admin

Block IP List

4 items found, displaying all items.1

Sr.No	IP Address	No Of Attack's
1	192.168.0.102	1
2	192.168.0.103	4
3	192.168.0.107	10
4	0:0:0:0:0:0:0:1	7

Export options: CSV | Excel | XML

Fig.7. Block IP List of Analysis Module

"Block IP List" interface page contain the attack history of the attacker .It contains IP address and number of attacks from each IP address.

VIII. CONCLUSION

In this paper we first identified various types of SQLIA's. Then we investigated on SQL injection and CSS attack detection and prevention techniques. SQL injection attacks modify the parameters of a web based application in order to alter SQL statements that are parsed to retrieve data from database. Here we have proposed an effective approach ie. A reverse proxy framework which is based on sanitization technique consisting of data cleansing and message digest algorithm. SQL injection preventer and cross site scripting preventer are able to protect web application against SQLIA and XSS attack. There are many types of defence techniques exist against this types of attacks, but we believe avoidance is the best solution .Thus reverse proxy provides a very secure and efficient service between client and web server and user's confidential information is also protected while performing online transactions.

IX.ACKNOWLEDGMENT

We are very much thankful to all authors mentioned in the references for making the resources available and all respected people who helped us in designing and development of the work. I am also thankful to my guide Prof.Vishal Mogal, for his support and guidance.

REFERENCES

1. Jose Fonseca,Macro Vieira,Henrique Madeira,"Evaluation of Web security Mechanisms using Vulnerability and attack injection",IEEE Transactions on dependable and secure computing, Vol 11, No.5,September/October 2014.
2. Indrani Balasundaram 1 Dr. E. Ramaraj, "An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service", IJCSNS International Journal of Computer Science and Network Security, January, 2011.
3. Stephen W. Boyd and Angelos D. Keromytis, "SQLrand: Preventing SQL Injection Attacks," Columbia University.
4. Shaimaa Ezzat Salama, Mohamed I. Marie, Laila M. El-Fangary & Yehia K. Helmy, "Web Anomaly Misuse Intrusion Detection Framework for SQL Injection Detection", International Journal of Advanced Computer Science and Applications, 2012.
5. Veera Venkateswaramma P, "An Effective Approach for Protecting Web from SQL Injection Attacks", International Journal of Scientific & Engineering Research, Volume 3, 2012.
6. Anyi Liu, Yi Yuan, Duminda Wijesekera, "SQLProb: A Proxy-based Architecture towards Preventing SQL Injection Attacks", SAC'09, Honolulu, Hawaii, U.S.A, 2009.
7. William G.J. Halfond and Alessandro Orso "Preventing SQL Injection Attacks Using AMNESIA," ICSE'06, Shanghai, China, 2006.
8. Zhendong Su, Gary Wassermann," The Essence of Command Injection Attacks in Web Applications", Charleston, South Carolina, USA, 2006.
9. Lwin Khin Shar and Hee Beng Kuan Tan"Defeating SQL Injection", Nanyang Technological University Singapore, 2013 IEEE.
10. Anil Kumar Mandapati, Adilakshmi. Yannam, "Web Application Vulnerability Analysis Using Robust SQL Injection Attacks", International Journal of Engineering Trends and Technology, 2012.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

11. Konstantinos Kemalis and Theodoros Tzouramanis, “SQL-IDS: A Specification-based Approach for SQL-Injection Detection”, Fortaleza, Ceará, Brazil, 2008.
12. Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti, “Using Parse Tree Validation to Prevent SQL Injection Attacks”, Lisbon Portuga , September-2005
13. RomilRawat, Shailendra Kumar Shrivastav, “SQL injection attack Detection using SVM”, International Journal of Computer Applications 2012.
14. Amirmohammad Sadeghian, Mazdak Zamani,Shahidan Abdullah “A Taxonomy of SQL Injection Attacks”International Conference on Information and Creative Multimedia 2013 IEEE.
15. A. Seavanrhi, Jayasree Devi, Sudha Reddy, A.Indira, V.Satish Kumar, “Detecting SQL Injections from Web Applications”, International Journal of Engineering Science and Advanced Technology, 2009, pp. 664-671.
16. Atefeh Tajpour,Maslin Massrum and Mohammad Zaman Heydari “Comparison of SQL injection detection and prevention Techniques” UCSI University,Kuala Lumpur,Malaysia ,2010 IEEE.
17. Naresh Duhan and Bharti Saneja, “A Two Tier Defense Against SQL Injection”Kurushetra University, 2014 IEEE.
18. Asagba P.O and Ogheneovo E.E, “A Proposed Architecture For Defending Against Command Injection Attack in a Distributed Network Environment”, Nigeria Computer Society, ITePED, 2011.