An Approach for Detecting and Preventing SQL Injection and Cross Site Scripting Attacks using Query sanitization with regular expression

[1]Monali Sachin Kawalkar, Dr. P. K. Butey[2]
[1]*Department of computer science, R.T.M.Nagpur University*
[2]*Head of the Department, Department of Computer Science Kamla Nehru, R.T.M Nagpur University*

Abstract *-- Because of digitalization and rapid growth in technology web applications are widely used like e-commerce, online payments, online banking, money transfer, social networking, etc. As web application interacts with database where critical information is stored over the network. The methodology used is Structure Query language (SQL) and Scripting language.OWASP[2] has released the latest version of "Top 10 Vulnerabilities" based on the previous incidents as well as on the risks associated with the Vulnerabilities. SQL Injection and Cross Site Scripting vulnerabilities are more prominent and harmful and have taken the highest Rank amongst the rest of the Top 10 OWASP Vulnerabilities. The SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The attack takes advantage of poor input validation in code and website administration. It allows attackers to obtain unauthorized access to the back-and database to change the intended application generated SQL queries. Cross Site Scripting is a most prevalent web application security issue. This occurs when application sends the user provided data to the web browser without validating or encoding the account. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites. In this paper present new Sql injection and Cross site scripting attacks defense approach. The paper identifies vulnerability attacks caused due to inputs performed by a user which are not properly validated in the web applications. In this approach we remove the attacks by using input query sanitization with the help of regular expression based database independent server side background service. This service will stop the attack before it affect the system and will provide a sanitized query to the system by classifying the input data into Sql or html input.*

Keywords- *OWASP – Open Web Application Security Project (OWASP) , SQL – Structure Query language, Web Application ,Detection and Prevention Techniques.XSS – Cross site scripting, Input query sanitization, DOM - Document Object Model.*

1Introduction

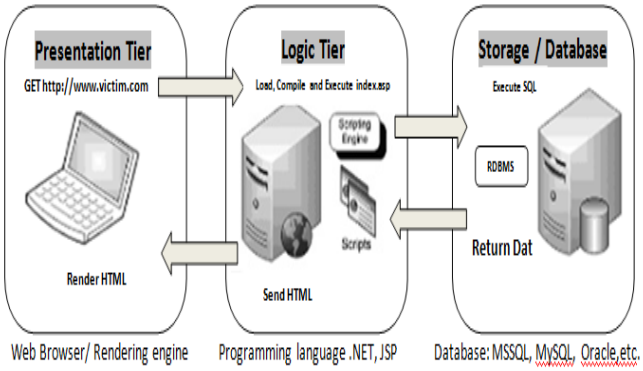
Web Applications are vulnerable to a variety of new security threats getting generated everyday by various sources, these applications which are hosted on Internet which is a widespread information infrastructure. Unaware of the Confidentiality, Integrity, Availability, Security and Privacy, the internet is becoming a repository of Business critical information. No matter which business, Information and data of organization is the most important business asset in today's environment, this can be achieved an appropriate level of Information security. Websites rely heavily on complex web applications to deliver different output or content to a wide variety of users according to set preferences and specific needs. This arms organizations with the ability to provide better value to their customers and prospects. However, the dynamic websites suffer from serious vulnerabilities rendering organizations helpless and committed to SQL injection attacks and cross site scripting attacks on their data. E-commerce sites are tricked by attackers and they lead into shipping goods for no charge, usernames and passwords have been cracked, and confidential and important credentials of users have been leaked. if there is no validation on the input of the application, then the malicious code can steal sessions, cookies, or inject and show private data for the user

1.1SQL injection Attack

Today most of the web applications are used multi-tier architecture. It includes with presentation layer i.e. front end, this is the topmost level of the application. This tier displays information related to such services as browsing, purchasing, and shopping cart contents. By outputting results the communication is done with the other tiers to the browser/client tier and all other tiers in the network. Application tier(Logic tier),this tier implements the software functionality by performing detailed processing and the data tier (Backend), this tier

consists of database servers, keeps data structured and answers to request from the application tiers. The user interface, functional process logic, data storage and access are developed and maintained as independent modules, most often on separate platforms in three tier client server architecture. SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to web applications underlying database and destroy functionality or confidentiality.[7] Figure 1 shows Simple Three Tier Architecture.

To gain access to the database content by unauthorized user, which is employed by a technique, known as SQL injection. Attacker would send specially framed input with SEL keywords embedded that would result in altering the semantic of the query. The concept of injection attack is to insert malicious code into a program so that result in change structure of SQL query. Such an attack may be performed by adding strings of malicious characters into data values in the form or argument values in the URL. The advantages of improper validation over input/output datais generally taken by injection attacks.

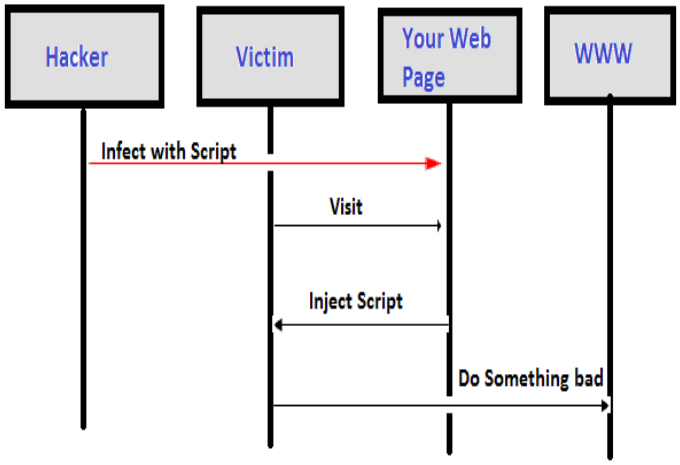


Simple Three Tier Architecture

Figure-1

1.2 Cross site scripting Attack

A cross-site scripting attack is a kind of attack on web applications in which attackers try to inject malicious scripts to perform malicious actions on trusted websites. In cross-site scripting, malicious code executes on the browser side and affects users. Cross-site scripting is also known as an XSS attack. A cross-site scripting attack occurs when a web application executes a script that the attacker supplied to end users. This flaw can be found anywhere in an application where user input has been taken but not properly encoded. If the input is not properly encoded and sanitized, this injected malicious script will be sent to users. And a browser has no way to know that it should not trust a script. When the browser executes the script, a malicious action is performed on the client side. Most of the times, XSS is used to steal cookies and steal session tokens of a valid user to perform session hijacking. Figure 2 shows a high level view of a typical XSS Attack.



A High Level View of a Typical XSS Attack

Figure 2.

2 Related Works

Existing system in practice used in development and test time to prevent or detect vulnerability attacks so that it improve programs for input validation vulnerability attacks can be reduced.Following section includes study of those techniques and also compare with our approach.

(A)Detection techniques for SQL injection Attacks- AMNESIA(Analysis and Monitoring for Neutralizing SQL Injection Attacks)

This is the most relevant detection technique proposed by Halfond et.al.[11], author suggested that AMNESIA is the effective SQLAs detection tool. It

is a model based techniques which combines both static analysis and dynamic analysis for preventing and detecting web application vulnerability at run time. Static phase is used to generate different type of query statements. Dynamic phase is used to interpret all queries before they are submitted to the database and validate each query against the statically built query models. AMENSIA technique stops all queries before they are sent to database and validates each query statement against the AMNESIA models. Queries that violate the model represent potential SQLs and thus prevented from executing on the database.

SQLGAURD [12][13]- In Userinput base model SQLGAURD method is useful. This method checked at runtime which is expressed as grammar that only accept legal queries.SQL Guard checks the structure of the query before and after the addition of user-input based on the model. In this approach developer should modify code to use a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generated query.

SQL Checker[12] – This model uses a secret key at runtime checking so security of the approach is dependent on attackers. In SQL Check, the model is specified independently by the developer. It is similar to SQL Gaurd by which Model checks as a grammar that only accept legal queries. In this case developer should have to modify code to use a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generates query.

Tautology Checker [14] – This method provides an analysis framework for security. It is a static analysis and automated reasoning performs for checking any tautology statement contains in coding. The major drawback of this tool is, it having limited scope. So this tool it is not useful as much.

CANDID [14]– In java programming CANDID tool is use as dynamically for program transformation. This tool dynamically mines the programmer-intended query structure on any input and detects attacks by comparing it against the structure of the actual query issued. For the detection of SQL injection attacks the CANDID's natural and simple approach turns out to be very powerful.

SQL-IDS [12] - Machine learning technique includes this method. It builds a model of typical queries and matches at run time that queries that does not match with original query treat as attacks. This technique detects attacks successfully, but it depends on training seriously.

SQL Prevent [12]-This technique is consists of an HTTP request interceptor. When the original data flow is modified SQL Prevent technique is deployed into a web server. The HTTP requests are saved into the current thread of local storage. The SQL statements are intercepts by SQL interceptor that are made by web application and pass them to the SQLIA detector module. Consequently, HTTP request from thread-local storage is fetched and checks to determine whether it contains an SQLIA. The malicious SQL statement would be prevented to be submitted to database, if it contains malicious data.

SQLR and [15] - According to the Keromytis and Boyd in SQLR and Proxy server is used between Client (Web server) and SQL server. They de-randomized queries received from the client and sent the request to the server. Portability and security are the advantages of this de-randomization framework.

(B)Prevention techniques for SQL Injection Attacks-

WAVES [16] - WAVES a black-box technique for testing web application for SQL injection vulnerability. This technique uses a Web crawler to identify all points in a Web application that can be used to inject SQLIAs. It target on a specified list of patterns and attack techniques. In WAVES , the application's response to the attacks is monitered by the waves and uses to improve its attack methodology it uses machine learning techniques.

JDBC Checker [13] –This technique is based on static analysis of web application that can reduce SQL injection vulnerabilities and detect type errors.It is uses for dynamically generated query string on basis of mismatching. As we know that most of the SQLIAs consist of syntactically and type correct queries so this technique would not catch more general forms of attacks.

SECURITY Fly [12]-Thisis implemented for java. As compare to other tool this checks string in place of character for any suspicious information and try to sanitize query strings. This tool has a drawback that is numeric fields cannot stop by this approach.

Difficulty of identifying all sources of user input is the main limitation of this approach.

SECURITY GATWAY [14] - It technique base on the filtering system that forces the input validation. By using Security Policy Descriptor Language (SPDL), developers provided specify transformation that is applied to the parameters of web application.

SQL DOM [12] - It is an object model for proposing a solution for building a secure communication environment for accessing relational databases from the OOP (Object-Oriented Programming) Languages. Due to this they mainly focus on identifying the obstacles in the interaction with the database via Call Level Interfaces.

WebSSARI [12]-Use for sanitizing input that passed through predefined set of filters. In this case static analysis to check taint flows against preconditions for sensitive functions. The drawback of this approach is that it is not necessary preconditions for sensitive function accurately expressed.

Similarly,various detection and prevention methods are being research and implemented in the past to secure web application from cross site scripting attacks.The related work includes Cross-site Scripting (XSS) Attack Detectionand Cross-site Scripting (XSS) Attack prevention techniques which are mainly based on static analysis work, dynamic analysis work, static and dynamic analysis, server side solution and client side solution.

In the area of static analysis [17]Y. W Huang, F. Yu, C. Hang, C. H. Tsai, D. Lee and S. Y. Kuo describe the use of bounded model checking (BMC) for verifying Web application code. Y. Huang, S. Huang, Lin, and Tsai use number of software-testing techniques. These techniques includes black-box testing, fault injection, and behavior monitoring to web application in order to work out the presence of vulnerabilities [18]. [19]A.S. Christensen, A. Møller, and M.I. Schwartzbach describe the analysis of string expression. For this they use Java programs and checking for errors in dynamically generated SQL queries. In Taint Propagation Analysis technique they use data flow analysis to track the behavior of information flow from source to sink[20,21]. D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel and G. Vigna[22] descibe a novel approach to analysis of the sanitization process in Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications. Wassermann and Su's in their recent work [23] describe Using of untrusted scripts to detect harmful script from user given data.

In the area of dynamic analysis work,Su and Wassermann in [24] describe a successful injection attack there is a change in the syntactical structure of the exploited entity.[25]E.Kirda et al developed Noxes which is the first client-side solution to mitigate cross-site scripting attacks. Noxes acts as a web proxy, called it as personal firewall.Browser-Enforced Embedded Policies Techniques by T.Jim, N.Swamy and M.Hicks [26] developed a mechanism that modifies the browser so that it can execute only filter content to prevent injected script code from running in browsers that view the site. Interpreter-based Approaches technique was introduced to track un-trusted data at the character level and for identifying vulnerabilities that use context-sensitive string. This approach described by T. Pietraszek and C. V. Berghe[27].

In the area of static and dynamic analysis,Lattice-based Approach described by [28]D. Balzarotti, M. Cova, V. V. Felmetsger and G. Vigna,described using WebSSARI which combines static and runtime features and find security vulnerability by applying static taint propagation analysis. WebSSARI targets cross site scripting.

In the area of server side solution and in the client side solution [29] Rattipong Putthacharoen and Pratheep Bunyatnoparat described protecting cookies from Cross Site Script attacks using Dynamic Cookies Rewriting technique. This is implemented in a web proxy where it will automatically rewrite the cookies that are sent back and forward between the users and the web applications. [30]Prithvi Bisht and Venkatakrishanan describes preventing mechanisms for cross site scripting attacks which is based on input validation that can effectively prevent XSS attacks on the server side. They also introduce several new XSS attacks and analyzed the reasons for the failure of filtering mechanisms in defending these attacks. They design XSS GUARD framework for preventing XSS

attacks on the server side. XSS GUARD works by dynamically learning the set of scripts that a web application intends to create for any HTML request. [31]N. Ikemiya and N. Hanakawa, describe client-side mechanism for detecting malicious Java Scripts. The system consists of a browser-embedded script auditing component, and IDS that processes the audit logs and compares them to signatures of known malicious behavior or attacks. By using Noxes Tool the client side cross site scripting is protected. [32] is a client-side Web-proxy that includes all Web traffic and serves as an application-level firewall. The approach works without attack-specific signatures.

3Proposed System

To handle SQL injection attack and Cross site scripting attack the following defenses are used for prevention. We are using Query sanitization with regular expression methodology to prevent or remove the SQL injections and XSS attacks which is fully database independent server side background service. This service we called as SQL injection and XSS removal service. This service will stop the attack before it affect the system and will provide a sanitize query to the system by classifying the input data into SQL or HTML input. Our proposed system is placed in between client and web server which is server side services.The general work of the system is as follows:

1) The client sends the request to server.
2) The request is redirected to method query sanitization with regular expression.
3) This method describes SQL injection attack and XSS attacks removal service, which uses Data classifier for checking the SQL statement and XSS from the URL.
4) If http request is SQL statement then method calls SQL injection removal service which sanitizes the query with regular expression and validates the request. These validated requests then send to the web application in the server.
5) If http request is XSS then method calls XSS attacks removal service, which sanitize the script with regular expression and validate the request. These validated requests then send to the web application in the server.
6) Else the request does not contain any malicious code, then access directly to web application server.
7) Depending on the validation results the filter on web application server decides whether to continue with the request or deny the request.

The following figure 3shows the system architecture for Query sanitization with regular expression.

3.1 Block diagram of System Architecture

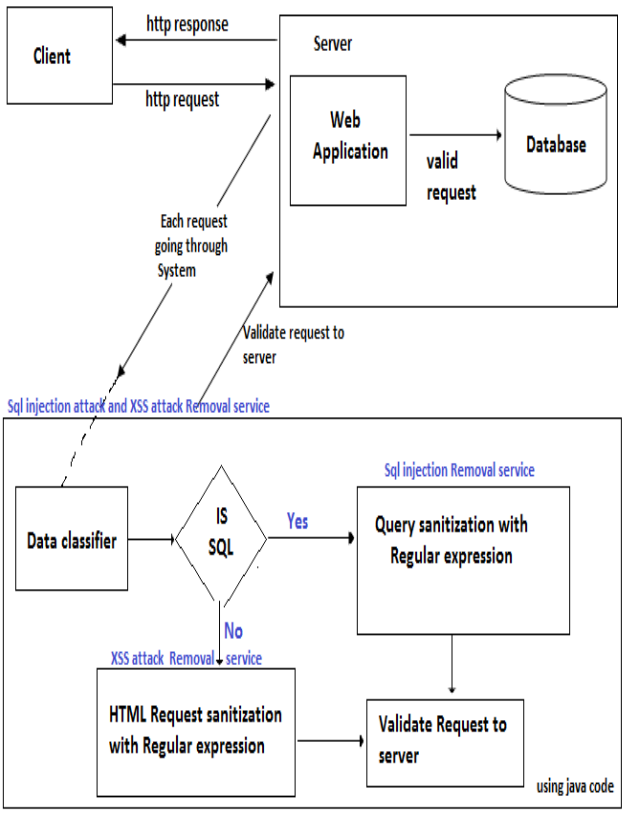


Figure 3 Proposed System Architecture (Query sanitization with regular expression)

3.2 Flowchart of the system

The following Figure 4 gives the flowchart of the proposed system.

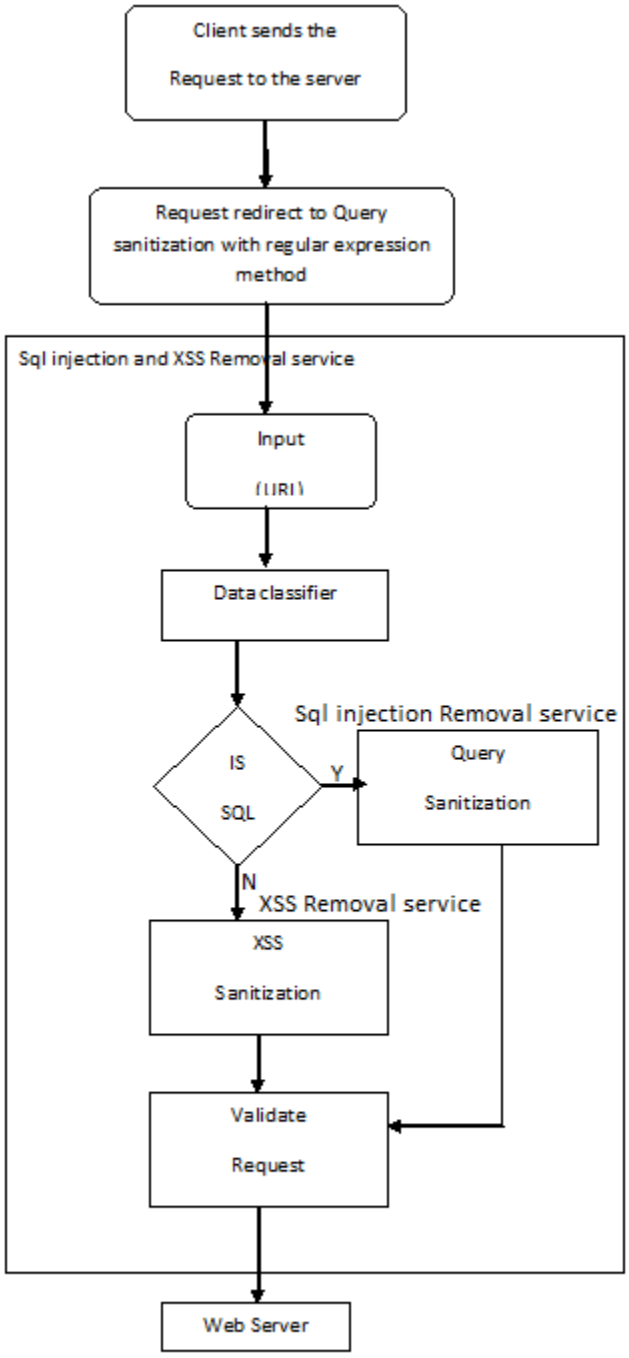


Figure 4 Flowchart of the system

3.3 Algorithm for database independent Sql injection attack and XSS attack removal using regular expression.

Steps:

1. Start the algorithm.
2. Accept user input in the form of any html text having scripts, tags, urls.
3. In Sql injection and XSS Removal services, data classifiers classify the sql statements and scripts.
   Check if (Sql)
   Call Sql injection Removal service
   Else
   Call XSS Removal service
4. Validated request send to web application server.
5. Repeat for each request.
6. End of the algorithm.

4. System Implementation

A web application utilizes web and browser technologies to perform tasks over a network using a web browser. The web applications are stored on the web servers, where all their data are stored.

1) This system implements the database independent SQL injection attack and XSS attack removal using regular expression algorithm using Java. Eclipse integrated development environment (IDE) runs the open source module. The application takes the input from the URL and then by using regular expression in Java data classifier classifies the SQL statement and script accordingly.
2) The sanitizing application is executed in the Eclipse ID for each request. When the redirected request from the server reaches the sanitizing application algorithm is triggered.
3) As a first step the algorithm checks whether it is a SQL query or script which is extracted from the URL. The SQL query and script are processed using search pattern. A sequence of character that forms a search pattern is called as regular expression. This search pattern can be used for text search and texts replace operations.
4) The URL is passed onto the signature check which uses the regular expression to validate the URL.
5) Some of the following checks are done on the URL extracted from the http request.
   I. Query delimeter (--)
   II. White spaces

III. Comment Delimeter (/**/)
IV. Scanning for the query with
V. Dropping Meta character like (;,',<, >, %, +)

6) The validated URL are then directed back to the server.
7) Depending on the validation results the filter on web application server decides whether to continue with the request or deny the request.

5. Evaluation

The system implements Query sanitization with regular expression method which is detection and prevention method that remove the SQL injection and XSS attacks successfully. For the evaluation expression we used the java code for creating java services. This method successfully tested on open source project. The following output screens Figure 5shows the response of the system when a malicious input is provided in the input form. Figure 6 and 7 shows outcome of malicious queries executed on application which sanitizes with our methodology i.e. using regular expression for removal of Sql injection attack and XSS attack on independent database.

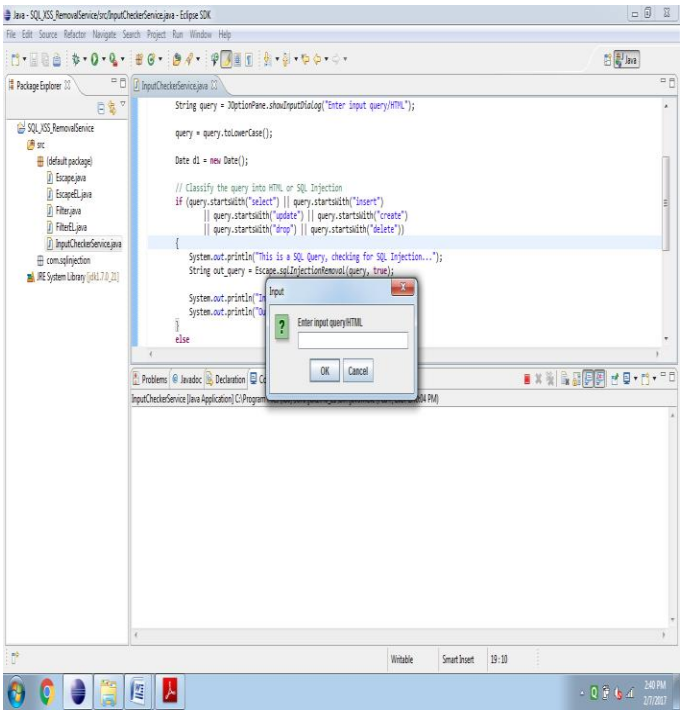


Figure 5 Malicious input provided to the application.

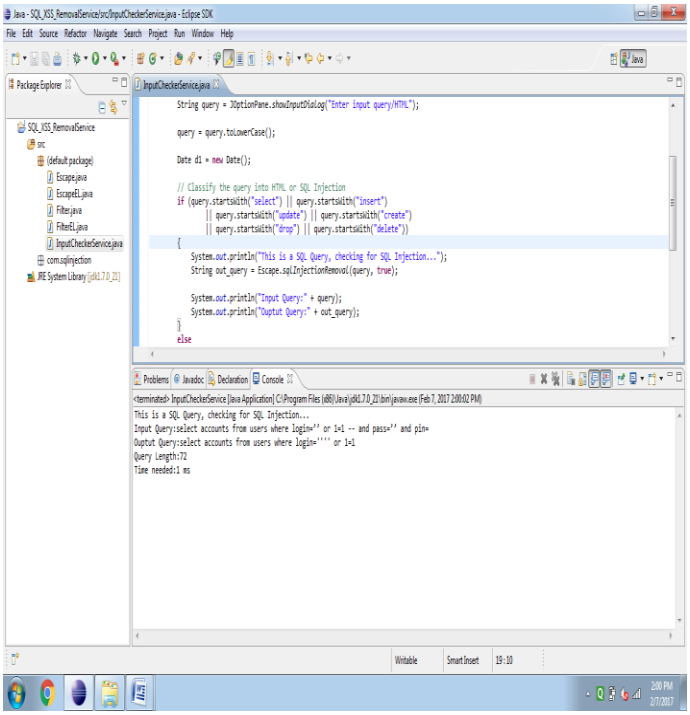


Figure 6 Sanitize the malicious input provided to the application.

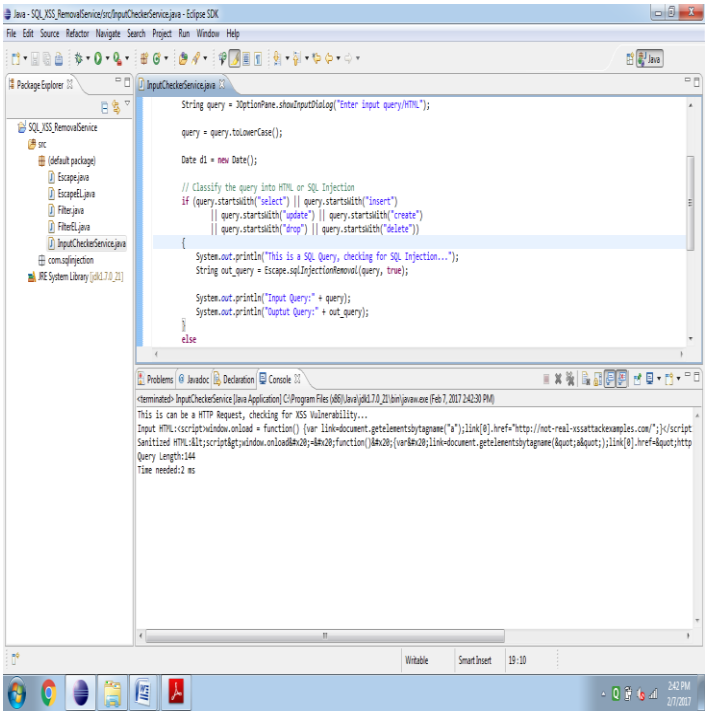


Figure 7 Sanitize the malicious input provided to the application.

6Analysis result

This section compares the detection rate of the proposed method with other researcher's method. Table 1 and 2 shows the comparison of SQL injection detection and prevention techniques with respect to attack type. With our proposed methodology it detects and prevents all type of attack successfully.We have analyzed the system and methodology that are used to control SQL injection attacks and XSS attacks which are shown in table3 and table4,describes the response time for the query executed irrespective of query length with accuracy.The time taken for the response with our system was noted in microseconds.

Table 1. Comparison of SQL injection detection techniques with respect to attack types

Attacks / Tools	Tautology	Piggy Baked	Ileagal Incorrect	Union	Alternate encoding	Timing attack	Blind attack	Stored procedure
AMNESIA	√	√	√	√	√	√	√	x
SQL GUARD	√	√	√	√	√	√	√	x
SQL Checker	√	√	√	√	√	√	√	√
TAUTOLOGY Checker	√	x	x	x	x	x	x	x
CANDID	√	x	x	x	x	x	x	x
SQL IDS	√	√	√	√	√	√	√	√
SQL Prevent	√	√	√	√	√	√	√	√
SQL RAND	√	√	√	√	√	√	√	x
Proposed method	√	√	√	√	√	√	√	√

Table 2. Comparison of SQL injection preventiontechniques with respect to attack types

Attacks / Tools	Tautology	Piggy Baked	Ileagal Incorrect	Union	Alternate encoding	Timing attack	Blind attack	Stored procedur
WAVES	0	0	0	0	0	0	0	0
JDBC Checker	0	0	0	0	0	0	0	0
SECURITY Fly	0	0	0	0	0	0	0	0
SECURITY Gateway	0	0	0	0	0	0	0	0
SQL DOM	√	√	√	√	√	√	√	x
WebSAARI	√	√	√	√	√	√	√	√
Proposed method	√	√	√	√	√	√	√	√

Table 3. Analysis of SQL injection

Sr. No	Query Length	Time Taken	Accuracy
1	72	1 ms	√
2	128	2 ms	√
3	92	1 ms	√
4	128	2 ms	√
5	121	2 ms	√
6	106	1 ms	√
7	153	2 ms	√
8	174	2 ms	√
9	195	2 ms	√
10	80	1 ms	√
11	75	1 ms	√
12	37	1 ms	√
13	63	1 ms	√
14	42	1 ms	√
15	44	2 ms	√
16	39	1 ms	√
17	55	1 ms	√
18	81	1 ms	√
19	57	1 ms	√
20	62	2 ms	√

Table 4. Analysis of XSS

Sr. No	Query Length	Time Taken	Accuracy
1	144	2 ms	√
2	61	1 ms	√
3	121	2ms	√
4	80	1ms	√
5	41	1ms	√
6	154	2ms	√
7	86	1ms	√
8	102	1 ms	√
9	88	2 ms	√
10	75	1 ms	√
11	89	1 ms	√
12	215	1 ms	√
13	28	1 ms	√
14	78	2 ms	√
15	56	1 ms	√
16	51	1 ms	√
17	61	1 ms	√
18	125	1 ms	√
19	53	1 ms	√
20	97	1 ms	√

Conclusion

This paper presents a survey on web application attacks i.e. types of vulnerabilities and security threats within the Web application(It can be e-commerce, social networking etc.). This paper proposes improved detection and prevention of input validation attack on web applications. Our proposed detection concept will help to detect and recognize XSS and SQL injection attacks and also sanitize the query for validation. It also expects that the concept will reduce the analysis time. Future work of our study will be the implementation of technique that uses method for detection and prevention of Input validation attacks like SQL injection and XSS on web application. Additionally this paper proposes improved and efficient tool that would provide web security.

References

[1] Www.OWASP.org/index.php/XSS_Prevention_Cheat_sheet
[2] OWASP Top Ten Project - http://www.owasp.org/index.php/Top_10
[3] OWASP Code Review Guide - http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
[4] OWASP Testing Guide - http://www.owasp.org/index.php/Testing_Guide
[5] https://www.acunetix.cz/websitesecurity/cross-site-scripting/
[6] "SQL Injection/Insertion Attacks". insecure.org.
[7] SQL Injection Attacks and Defense(Book) - Justin Clarke
[8] Cross Site Scripting Wikipedia, http://en.wikipedia.org/wiki/Cross-site_scripting
[9] Cross site scripting ,accunetix,http://www.acunetix.com/websitesecurity/cross-site-scripting/
[10] Cross site scripting, Secure web development, ,http://hwang.cisdept.csupomona.edu/swanew/Code.aspx?m=XSS
[11] W. G. J. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," presented at the Proceedings of the 28th international conference on Software engineering,Shanghai, China, 2006.
[12] Sh. Bojken, A. Shqiponja, A. Marin, and Xh. Aleksander,"Protection of Personal Data in Information Systems",International Journal of Computer Science, Vol. 10, No. 2,July 2013, ISSN (Online): 1694-0784.
[13] Srinivas Avireddy, Varalaxhmi perumal, Narayan Gowraj, Ram Srivastava Kannan"Random4: An Application Specific Randomized Encryption Algorithm to prevent SQL Injection" 11th International conference on trust, Security and privacy in computing and communications IEEE 2012.
[14] A Survey on Detection and Prevention Techniques of SQL Injection by Harish Dehariya
[15] S. W. Boyd and A. D. Keromytis. "SQLrand: Preventing SQL Injection Attacks", In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.
[16] Y. Huang, F. Yu, C. Yang, C. H. Tsai, D. T. Lee, and S. Y. Ku. "Securing Web Application Code by Static Analysis and Runtime Protection", In Proceedings of the 12th International Word Wide Web Conference, May 2004.
[17] Y. W Huang, F. Yu, C. Hang, C. H. Tsai, D. Lee and S. Y. Kuo, "Verifying Web Application using Bounded Model Checking," In Proceedings of the International Conference on Dependable Systems and Networks, (2004).
[18] Y.-W. Huang, S.-K. Huang, T.-P. Lin and C.-H. Tsai, "Web application security assessment by fault injection and Behavior Monitoring," In Proceeding of the 12th international conference on World Wide Web, ACM, New York, NY, USA, (2003), pp.148-159.
[19]]A. S. Christensen, A. Mǿller and M. I. Schwartzbach, "Precise analysis of string expression", In proceedings of the 10th international static analysis symposium, LNCS, Springer-Verlag, vol. 2694, pp. 1-18.
[20] V.B. Livshits and M. S. Lam, "Finding security errors in Java programs with static analysis," In proceedings of the 14th Usenix security symposium, (2005)August, pp. 271-286.
[21] N. Jovanovic, C. Kruegel and E. Kirda, "Precise alias analysis for syntactic detection of web application vulnerabilities," In ACM SIGPLAN Workshop on Programming Languages and Analysis for security, Ottowa, Canada, (2006)June.
[22] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel and G. Vigna, "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," In IEEE symposium on Security and Privacy, (2008).
[23] G. Wassermann and Z. Su, "Static detection of cross-site Scripting vulnerabilities," In Proceeding of the 30th International Conference on Software Engineering, (2008)May.
[24] Z. Su and G. Wassermann, "The essence of command Injection Attacks in Web Applications," In Proceeding of the 33rd Annual Symposium on Principles of Programming Languages, USA: ACM, **(2006)** January, pp. 372-382.
[25]] E. Kirda et al., "Client-Side Cross-Site Scripting Protection," Computers & Security,"Proc of 21st ACM Symposium on Applied Computing,Oct. 2009, pp. 592-604.
[26]]T. Jim, N. Swamy and M. Hicks, "BEEP: Browser-Enforced Embedded Policies," In Proceedings of the 16th International World Wide Web Conference, ACM, (2007), pp. 601-610.
[27] T. Pietraszek and C. V. Berghe, "Defending against Injection Attacks through Context-Sensitive String Evaluation", In Proceeding of the 8th International Symposium on Recent Advance in Intrusion Detection (RAID), (2005)September.
[28] D. Balzarotti, M. Cova, V. V. Felmetsger and G. Vigna, "Multi-Module Vulnerability Analysis of Web-based Applications," In proceeding of 14th ACM Conference on Computer and Communications Security,Alexandria, Virginia, USA, (2007)October.
[29] R.Putthacharoen and P.Bunyatnoparat," Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewritng Technique,"Proc. of IEEE 13th International Conference on Advanced Communication Technology, Feb 2011,pp. 1090-1094.
[30] P. Bisht and V. N. Venkatakrishnan, "XSS-GUARD: Precise dynamic prevention of Cross-Site Scripting Attacks," In Proceeding of 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS, vol. 5137, **(2008)**, pp. 23-43.
[31] N. Ikemiya and N. Hanakawa, "A New Web Browser Including A Transferable Function to Ajax Codes", In Proceedings of 21st IEEE/ACM International Conference on Automated Software Engineering (ASE '06), Tokyo, Japan, **(2006)** September, pp. 351-352.
[32] E. Kirda et al., "Client-Side Cross-Site Scripting Protection," Computers & Security,"Proc of 21st ACM Symposium on Applied Computing,Oct. 2009, pp. 592-604.