International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT22816

XSS and SQL Injection Detection and Prevention Techniques (A Review)

[1]Bhanwarlal, [2]Irfan Khan

[1]PG Scholar, Department of Computer Science and Engineering, Shekhawati Institute of Engineering and Technology, Sikar, Rajasthan, India

[2]Assistant Professor, Department of Computer Science and Engineering, Shekhawati Institute of Engineering and Technology, Sikar, Rajasthan, India

Article Info
Volume 8, Issue 1
Page Number : 53-60

Publication Issue :
January-February-2022

Article History
Accepted : 02 Jan 2022
Published : 13 Jan 2022

ABSTRACT

In modern times every human being rely upon the internet for their scant to hefty needs as internet offers vast amount of information to users, so it's availability to users is indispensable. Major objectives of security are availability, integrity and confidentiality. Cross Site Scripting (XSS) and SQL Injection Attack (SQLIA) is a generic and critical security issue towards to the web application and database security. In general, not well validated and verified web applications are highly prone and vulnerable by the attackers. Due to the creative and dynamic XSS and SQLIA methods and techniques, users can save their valuable, integral and confidential data in the web site to save their market stability towards their self as well as social enrichment.

Many tools and techniques are addressed to the references regarding the XSS and SQL Injection issues, but we are present and used pattern matching techniques in SQL statements to implement the SQLIA and XSS in web application. At the outset pattern matching algorithm is used and gets better solution towards on implementation of SQLIA and XSS attacks and preventions.

Keywords - SQLIA, SQL queries, XSS, Cross site Scripting, web application, asp.net., Security, Internet, Server.

I. INTRODUCTION

Web services now use online apps to implement the tools and use the web platform to become an innovative solution for companies with software applications. It helps the development of universal applications that can be conveniently accessed by millions of users. World Wide Web (WWW) has been a great advancement but internet attacks have risen at the same time. There will be millions of security vulnerabilities occurring day after day. However, according to Akamai [1], the volume of internet device attacks rose 69 percent in the third quarter of 2017 compared with Q3 2016. Although SQL injection and local File Intrusion attacks accounted for 85% of all these assaults, XSS was just

9%. For Verizon's 2017 Data Breach Study [2], just 15.4 percent of reported web application assault cases, while web application assaults suffered 29.5 percent of the breaches. SQL Injection is the most common online attack. Figure 1 shows a recent article on SQL Injection.

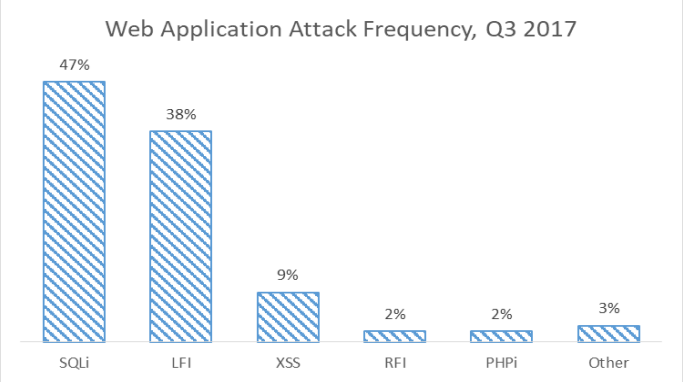

Figure 1.1: A chart representing XSS and Sql Injection Attack Frequency

According to studies carried out by U.S. cloud service provider Akamai, who discovered in his Internet State of Report [13] that SQL injection and Local Filer Inclusion assaults accounted for more than 85% of the assault vectors identified. SQL injection attacks accounted for 65 per cent of web-based attack vectors from November 2017 to March 2019.

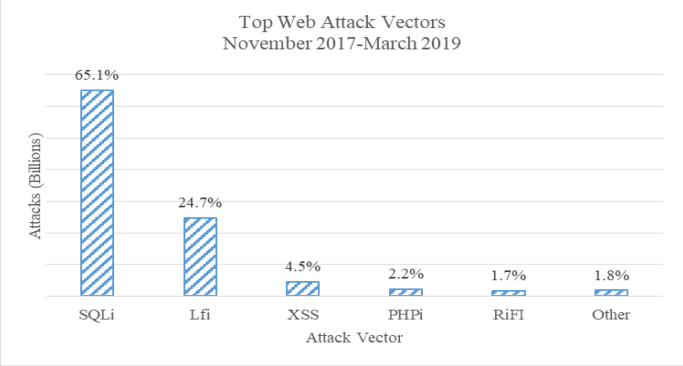

Figure 1.2: Representing Web Attacks

Akamai registered cyber alerts of just under four billion (3,993) threats over a 17-month span, of which just 1.23 billion happened in the first half of 2019.

The primary concern of internet is security against repudiation, integrity, confidentiality and availability of information. Attacks obstruct the legitimate user to access the network services and allow attacker to access the all leverage of services. It can be conducted by using single machine or multiple machines named as zombies.

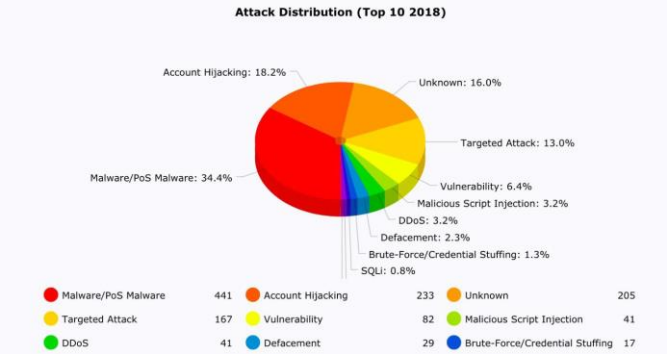

Figure 1.3: Top internet attack distribution data

Web App or Web applications are the tools to share and display their personnel or self-valuable information to access in worldwide network from anywhere. Regarding the usage of the web applications have many benefits, out of which some of the issues are much more risky. Now a day's information as well as network security is a big concern, where users can face many types of attack, out of which one simple and common category is SQLIA and XSS. As we know that Web applications are one such tool to access and transfer various self - information across the world with internet facility. Meanwhile some creative and dynamic hackers are also waiting and accomplish to hack this precious information from internet and breach the privacy of users. So that attackers have used many techniques, out of which SQL Injection Attack (SQLIA) and XSS are the major and common attacks performed by the attacker. As per the internet and network security concern SQLIA and XSS are one of the serious threats in the web application. XSS and SQLIA are considered to be in top ten vulnerability for web application usage. Now a day's SQLIA and XSS will be the easiest way to hack or attack the Web applications with various servers in using in World Wide Web protocol, i.e. if the web applications is coded very Poor in programming language or if the system files are not

uploaded in the system, then it dam sure that, such web applications are very easy to hack by the attackers.

1.1 Introduction to SQL Injection:

Generally speaking, people say that SQL injection isn't new to them and they know about this concept but the reality is completely different from it and they've either read about it or encountered it by superficial scenarios. SQL injection is one of the most damaging flaws and can have a very significant effect on business since it will reveal any confidential information contained in an application's database, including valuable information such as usernames , passwords, identities, emails, telephone numbers, credit and debit card details.

The key to understanding SQL Injection is in its name: SQL + Injection. The word "injection" here doesn't have any medical connotations, but rather is the usage of the verb "inject." Together, these two words convey the idea of putting SQL into a web application. Putting SQL into a web application . . . hmmm . . . Isn't that what we're doing anyway? Yes, but we don't want an attacker to drive our database Web App or Web applications are the tools to share and display their personnel or self-valuable information to access in worldwide network from anywhere. Regarding the usage of the web applications have many benefits, out of which some of the issues are much more risky. Now a day's information as well as network security is a big concern, where users can face many types of attack, out of which one simple and common category is SQLIA.As we know that Web applications are one such tool to access and transfer various self - information across the world with internet facility. Meanwhile some creative and dynamic hackers are also waiting and accomplish to hack this precious information from internet and breach the privacy of users. So that attackers have use many techniques, out of which SQL Injection Attack (SQLIA) is the major and common attacks performed by the attacker. As per the internet and network security concern SQLIA is one of the serious threats in the web application and SQLIA is considered to be in top ten vulnerability for web application usage. Now a day's SQLIA will be the easiest way to hack or attack the Web applications with various servers in using in World Wide Web protocol, i.e. if the web applications is coded very Poor in programming language or if the system files are not uploaded in the system, then it dam sure that, such web applications are very easy to hack by the attackers. It has been found that to detect and prevent SQLIA tour system, we must use some efficient algorithm to safeguard our personal data, in this paper we analyze and use pattern matching algorithm to detect and prevent SQLIA. Naive String pattern matching algorithm is used and tested efficiently in SQLIA.

1.2 Introduction to Cross Site Scripting

Cross-Site Scripting is a kind of security exploit in which the attacker inserts malicious code of his choice (mostly script) into a web page or a database without the user's knowledge. XSS in itself is a threat which is brought by the internet security weaknesses of client-side scripting languages, with HTML and JavaScript (others being VBScript, ActiveX, HTML, or Flash) as the prime culprits for this exploit.

We can categorize XSS as follows:

- Reflected (when malicious code goes from the user's browser to the server and comes back from server)
- Persistent (when code remains stored somewhere, example - code stored in a database and executed on the client browser over and over, which makes it more dangerous).
- DOM based XSS attack (both reflected and persistent can fall in this category, attacker can

manipulate DOM elements and can use DOM data).

II. HISTORY OF XSS and SQLI ATTACKS

Several solutions that mitigate the risk posed by SQL Injection attacks have already been proposed [1], [7]–[9]. All of these solutions have been successful in mitigating SQL Injection attacks. However, none of these solutions address the actual SQL injection attack that exists in the source code. A common way to remove SQL injection vulnerability is to separate the SQL structure from the SQL input by using prepared statements.

Ruse et al.'s Approach, propose a technique that uses automatic test case generation to detect SQL Injection Vulnerabilities. The main idea behind this framework is based on creating a specific model that deals with SQL queries automatically. Adding to that, the approach identifies the relationship (dependency) between sub-queries. Based on the results, the methodology is shown to be able to specifically identify the causal set and obtain 85% and 69% reduction respectively while experimenting on few sample examples.

SAFELI et al.'s proposes a Static Analysis Framework in order to detect SQL Injection Vulnerabilities. SAFELI framework aims at identifying the SQL Injection attacks during the compile-time. This static analysis tool has two main advantages. Firstly, it does a White-box Static Analysis and secondly, it uses a Hybrid-Constraint Solver. For the White-box Static Analysis, the proposed approach considers the byte-code and deals mainly with strings. For the Hybrid-Constraint Solver, the method implements an efficient string analysis tool which is able to deal with Boolean, integer and string variables.

Ali et al.'s Scheme, adopts the hash value approach to further improve the user authentication mechanism. They use the user name and password hash values SQLIPA (SQL Injection Protector for Authentication) prototype was developed in order to test the framework. The user name and password hash values are created and calculated at runtime for the first time the particular user account is created.

Thomas et al.'s Scheme, suggest an automated prepared statement generation algorithm to remove SQL Injection Vulnerabilities. They implement their research work using four open source projects namely: (i) Nettrust, (ii) ITrust, (iii) WebGoat, and (iv) Roller. Based on the experimental results, their prepared statement code was able to successfully replace 94% of the SQLIVs in four open source projects. Dynamic Candidate Evaluations Approach

Bisht et al.'s propose CANDID. It is a Dynamic Candidate Evaluations method for automatic prevention of SQL Injection attacks. This framework dynamically extracts the query structures from every SQL query location which are intended by the developer (programmer). Hence, it solves the issue of manually modifying the application to create the prepared statements.

Haixia and Zhihong's et. al's Database Security Testing Scheme, propose a secure database testing design for Web applications. They suggest a few things; firstly, detection of potential input points of SQL Injection; secondly, generation of test cases automatically, then finally finding the database vulnerability by running the test cases to make a simulation attack to an application. The proposed methodology is shown to be efficient as it was able to detect the input points of SQL Injection exactly and on time as the authors expected. However, after analyzing the scheme, we find that the approach is not a complete solution but rather it needs additional improvements in two main aspects: the detection capability and the development of the attack rule library.

Swaddler et. al's, analyzes the internal state of a web application. It works based on both single and multiple variables and shows an impressive way against complex attacks to web applications. First the approach describes the normal values for the application's state variables in critical points of the application's components. Then, during the detection phase, it monitors the application's execution to identify abnormal states.

Stephen et. al's. proposed a prepared statement replacement algorithm and a corresponding tool for automated fix generation.

Cristian et.al's. presented a hybrid approach based on the Adaptive Intelligent Intrusion Detector Agent (AIIDASQL) for the detection of SQL injection attacks. The AIIDA-SQL agent incorporates a Case-Based Reasoning (CBR) engine which is equipped with learning and adaptation capabilities for the classification of SQL queries and detection of malicious user requests. To carry out the tasks of attack classification and detection, the agent incorporates advanced algorithms in the reason-ing cycle stages. Concretely, an innovative classification model based on a mixture of an Artificial Neuronal Network together with a Support Vector Machine is applied in the reuse stage of the CBR cycle. This allowed classification of SQL queries.

Michelle et. al's proposed a technique that is based on automatically developing a model for a SQL query such that the model captures the dependencies between various components (sub-queries) of the query. The authors analyzed the model using CREST test-case generator and identify the conditions under which the query corresponding to the model is deemed vulnerable. The authors further analyzed the obtained condition set to identify its subset; this subset being referred to as the causal set of the vulnerability. The technique proposed by the authors considers the semantics of the query conditions, i.e., the relationship between the conditions, and as such complements the existing techniques which only rely on syntactic structure of the SQL query. In short, the technique proposed by the authors can detect vulnerabilities in nested SQL queries.

Su and Wassermann et. al's develop a formal model for command injection attacks and apply a syntactic criterion to filter out malicious dynamic content. Applications of taint checking to server programs that generate content to ensure that untrustworthy input does not flow to vulnerable application components have also been explored UserCSP is a Mozilla tool that allows security savvy users to specify and en-force content security policy to protect themselves from cross-site scripting attacks. The tool automatically infers content security policies for the websites user visits and enforces them to protect users from XSS attacks. Other solutions need browser modifications to identify untrusted or malicious scripts from trusted scripts.

MashupOS [31] makes the browser a multi-principal operating system for Web applications. BEEP [13] lets Web sites restrict the scripts that run in each of their pages. ConScript [14] enforces application-specified security policies. OMash [3] restricts communication to public interfaces declared by each page.

Kailas et. al's proposed a solution to isolate untrusted scripts included in web applications from the trusted scripts. It allows isolation of Javascript context for scripts from different origins. In addition, it also provides different privileges of read and write to scripts running in isolated Javascript contexts. BrowserShield [24] propose to defeat JavaScript-based attacks by rewriting scripts according to a security policy prior to executing them in the browser. In BrowserShield, the rewriting process inserts trusted JavaScript functions to mediate access to the document tree by untrusted scripts.

Jackson et al's describe several unexpected repositories of private information in the browsers cache that could be stolen by XSS attacks. They advocate applying a refinement of the same-origin policy [15] to cover aspects of browser state that extend beyond cookies. By allowing the server to explicitly specify the scripts that it intentionally includes in the document, our approach can also be thought of as an extension of the same-origin policy.

III. SQLI and XSS ATTACKS

We observed that the aim of content injection attack to gain illegal access to user data. The Structural Query Language Injection (SQLI) attack occurs when an attacker changes the logic, semantics or syntax of a SQL query by inserting new SQL keywords or operators. SQL Injection attack is a class of content injection attacks that occurs when there is no input validation mechanism deployed by web developers in the web application. In cross-site scripting attack, the attackers fold malicious content into the content being delivered from the compromised site. When the resulting combined content arrives at the client-side web browser, it has all been delivered from the trusted source, and thus operates under the permissions granted to that system. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, session cookies, and a variety of other information maintained by the browser on behalf of the user [32]. The successful XSS attack is a result of lack to provide input validation in the web application by the developers. Too many existing techniques are either not publicly available or are difficult to adopt. Readily available tools would motivate more developers to combat content injection attacks. Developer's unawareness of security mechanisms and content injection sanitization can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover. Therefore, it is important to provide a solution that protects web applications from SQLI and XSS attacks. This paper performs the survey of various techniques proposed to protect web applications from these attacks.

IV. TECHNIQUES TO GENERATE XSS and SQLI ATTACK

4.1 Number of Attacks Sent

We generate an attack the number of attack packets were sent to victim while generating DDoS attack. Denoted the time at which attack packet were sent to server from different zombie machines and attack data were save into the text file for counting the number of attacks sent to server. It is simply stored into the text file and by using these files we got the number of attacks sent to the server as by counting number of lines in data file. Let us denote the number of attack packet sent to server is $P_{success}$.

For experimental purpose, data is collected by sending packets to server at start time $T_x$by all the machines andafter error time $T_y$machines get error message from server and server stops servicing them.Based upon the gathered information we measure performance of server and clients. Approx. 60 packets per second are delivered to the server and number of attack per second is evaluated using three attribute start time of packet sent, error time and number of packet successful packet, after total attack at instant is computed using attack per second (Y).

4.2 Attacks per Second

Attacks per second are evaluated using start time (T_x), error time (T_y) and number of attack packets sent ($P_{success}$)

Start time of packet sent = T_x

Error time = T_y

Number of packet sent = $P_{success}$

$$Attack\ per\ second(Y) = P_{success}/T_y - T_x$$

Using above formula attacks per secondd for first client is calculated as follows: start time is 19:28:38

and error time is 19:29:43 thus the difference between T_y-$T_x$is 65 seconds and number of packets$P_{success}$is 4152. The value of attacks per second Y calculated is 4152/65 = 63.88s and so on.

4.3 Total attacks at instant

The value of total attacks at instant is computed using the value of attacks per second (Y) i.e.

Y_n, Y_{n+1}, Y_{n+2}……. Where n is number of client machine.

$$\sum_{i=1}^{n} Y_n = X_i$$

Where X=Total attacks at instant.

For first machine total attack packets/sec is 63.88 and for second machine is obtained by adding the first and second machine data attacks per second and for third machine by adding machine first to third machine attacks per second and so on.

4.4 Server performance (Z)

At last using the computed value of total attacks at instant and attacks per second, server performance is evaluated

X = Total attacks at instant

Y = Attacks per Second

Z = Server Performance

$$Z = Maximum\ Attack\ per\ Second / Total\ Attack\ at\ Instant * 100$$

Or

$$Z = \frac{Y_{max}}{X_1} * 100$$

V. CONCLUSION AND FUTURE WORK

Traditionally, content injection was limited to personal computing environments. However, the increasing use of smart phones, tablets, and other portable devices has extended this problem to mobile and cloud computing environments, where vulnerabilities could spread much faster and become much easier to exploit. In this paper, we presented a survey of SQL injection and Cross-site scripting prevention research. This paper analyzed important aspects in content security systems. This survey paper serves as a guideline for researchers who are new to web security and want to contribute to this research area.

VI. REFERENCES

[1]. G. Buehrer, B.W. Weide, and P.A.G. Sivilotti. “Using parse tree validation to prevent sql injection attacks”. In Proceedings of the 5th International Workshop on Software Engineering and Middleware.

[2]. CGIsecurity. The cross-site scripting (xss) faq.http://www.cgisecurity.com/xss-faq.html.

[3]. S. Crites, F. Hsu, and H. Chen. Omash: “Enabling secure web mashups via object abstractions”. In Proceedings of the International Conference on Computer and Communications Security (CCS), 2008.

[4]. Xinshu Dong, Kailas Patil, Xuhui Liu, Jian Mao, and Zhenkai Liang. “An entensible security framework in web browsers”. Technical Report TR-SEC-2012-01, Systems Security Group, School of Computing, National University of Singapore, 2012.

[5]. Xinshu Dong,Kailas Patil, Jian Mao, and Zenkai Liang. “A comprehensive client-side behavior model for diagnosing attacks in ajax applications”. In proceedings of the 18th International Conference on Engineering of Complex Computer systems (ICECSS).

[6]. Dennis Fisher. Persistent XSS bug on twitter exploited by worm http://threatpost.com/en us/blogs/persistent-xssbug-twitter-being-exploited-092110

[7]. W.G.J.Halfond and A. Orso. “Amnesia: analysis and monitoring for neutralizing sql-injection attacks”. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering.

[8]. W.G.J. Halfond and A. Orso. "Combining static analysis and runtime monitoring to counter sql-injection attacks".In Proceed-ings of the Third International Workshop on Dynamic Analysis.

[9]. W.G.J. Halfond, A. Orso, and P. Manolios. "Using positive tainting and syntax-aware evaluation to counter sql-injection at-tacks". In Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[10]. Yichen Xie and Alex Aiken. "Static detection of security vulnerabilities in scripting languages". In Proceedings of the USENIX Security Symposium.

[11]. Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. "Protecting browser state from web privacy attacks". In Proceedings of the International Conference on World Wide Web (WWW).

[12]. Patil Kailas, Dong Xinshu, Li Xiaolei, Liang Zhenkai, and Jiang Xuxian. "Towards fine-grained access control in javascript contexts". In Proceedings of the International Conference on Distributed Computing Systems.

[13]. Ziqing Mao, Ninghui Li, and Ian Molloy. "Defeating cross-site request forgery attacks with browser-enforced authenticity protection". In Financial Cryptography and Data Security, 13th International Conference.

[14]. Leo A. Meyerovich and Benjamin Livshits. "ConScript: Specifying and enforcing fine-grained security policies for javascript in the browser". In Proceedings of the IEEE Symposium on Security and Privacy (IEEE S & P).

[15]. Mozilla Same origin policy for javascript. https://developer.mozilla.org/En/Same_origin_policy_for _javascript.

[16]. The clickjacking meets xss: a state of art. http://www.milw0rm.com/papers/265.

[17]. Anh Nguyen-tuong, Salvatore Guarnieri, Doug Greene, Jeff Shirley, and David Evans. "Automatically hardening web applications using precise tainting". In Proceeding of the 20th IFIP International Information Security Conference.

[18]. National Institute of standards and technology. National vulnerability database (nvd) http://web.nvd.nist.gov/view/vuln/search

[19]. Kailas Patil Ensuring session integrity in the browser environment http://scholarbank.nus.edu.sg/bitstream/handle/10635/49161/ThesisHT080141L.pdf?sequence=1.

[20]. Kailas Patil, Tanvi Vyas, Fredrik Braun, and Mark Goodwin. "Usercsp- user specified content security policies". SOUPS'13 POSTER

Cite this article as :

Bhanwarlal, Irfan Khan, "XSS and SQL Injection Detection and Prevention Techniques (A Review) ", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 1, pp. 53-60, January-February 2022. Available at doi : https://doi.org/10.32628/CSEIT22816
Journal URL : https://ijsrcseit.com/CSEIT22816