Analysis of Tools against Vulnerabilities in Web Applications

Sudhir Thakur, M. Tech. Scholar[*], Dr. Mrs. Poonam Sinha[#]
Dept. of IT, BUIT, Barkatullah University, Bhopal (M. P.)
[*]Sudhir_thakur0780@yahoo.co.in, [#]poonambuit@yahoo.com

***Abstract*—Now a day most of our activities including daily activities are being completed by internet and more specifically by using web applications. The hackers and attackers attack on the web applications for their financial benefit or for bad thinking or just for fun. For this they attack by SQL Injection, Cross-site Scripting and several other techniques. Research says that around85% of web applications are vulnerable to SQL Injection (SQLI) and Cross-site Scripting (XSS) attacks. Many tools and techniques have been developed to prevent and detect both above mentioned attacks. In this paper we will analyze these tools and their strength and weaknesses.**

***Keywords*—Web applications, Vulnerable, SQL Injection, Cross-site Scripting.**

I. INTRODUCTION

Two types of attacks are most common and dangerous. SQL Injection attack occurs when the attacker is able to insert SQL strings into a query by changing data input into an application. SQLIA is among the most common database attacks which try to access the sensitive and important data directly. Whereas XSS attack occurs when a user clicks on modified malicious URLs or visit an infected page.

The malicious script is executed on client's web server. XSS attack results in cookie theft, modifying user's account information and its privileges and much more. The XSS attack is similar to SQLI as they follow the same attacking style of code injection, but they are different from each other.

The SQLI attack performs to access and manipulate the application's database whereas in XSS attack, the attacker focuses on injecting malicious/harmful code to application user. Previous approaches identifying against SQLI and Cross Site Scripting vulnerabilities and preventing exploits include static analysis, defensive coding, test generation and dynamic monitoring.[6]

Each of these approaches has its own merits, but there are possibilities for improvement. Static analysis tools can produce false warnings and do not create concrete examples of inputs that exploit the vulnerabilities.

Defensive coding is error-prone and requires rewriting existing software to use safe libraries, so this is a big task. Black-box test generation does not take advantage of the application's internals, whereas previous White- box techniques have not been shown to discover unknown vulnerabilities. Dynamic monitoring tools outcomes in runtime overhead on the running application and do not detect vulnerabilities until the code has been deployed.

II. BACKGROUND

The SQL injection attack is used to exploit the web application remotely without any application or database authentication. By sending modified input a hacker can change the SQL statement structure and execute arbitrary SQL commands on the vulnerable piece of code. Consider the following username and password example, to login the web site, the user inputs his username and password, by clicking on the submit button the following SQL query is generated:-

SELECT * FROM employeetbl WHERE eid ='0011' and password = '112233'

The query will be executed when user input the following password:

'Or5=5 –

The SQL query will become:

***SELECT * FROM employeetbl WHERE eid='0011' and password = '112233' or 5=5 --'*.[3]**

The query will return all rows from the database regardless of whether eid='0011' and password='112233'.This is because the '5'='5'[10] will always return true result and OR statement appended to the WHERE clause. Therefore, the query will return a non-empty result set without any error. SQL injection problem can be solved by checking all SQL statements before sending them to the database. There are various SQL injection prevention and detection techniques developed [2]. In this paper we will review about these techniques with their strength and weaknesses.

III. SQL INJECTION ATTACK TYPES

There are different methods of attacks that depending on the goal of attacker are performed together or sequentially. For a successful SQLIA the attacker should append a syntactically correct command to the original

SQL query. Now the following classification of SQLIAs in accordance to the Halfond, Viegas and Orso researcher will be presented.

a. Illegal/Logically Incorrect Queries: Finding vulnerabilities through error messages.
b. Piggy-backed Queries: In this type of attack, the original query is appended by the query delimiter such as";" to append extra query.
c. Tautologies: the conditional query statement is evaluated always true like exploiting vulnerabilities in the database using WHERE clause.
d. Union Query: the attacker use the SQL tokens with the word UNION to injected query and then get data about other tables from the application.
e. Inference: By this type of attack, intruders change the behavior of a database or application.
f. Blind injection: stealing data by asking a series of True False questions through SQL statements.
g. Timing attacks: By observing timing delays in the database's responses.

IV. SQL INJECTION DETECTION AND PREVENTION TECHNIQUES

We discussed about various types of SQL Injection attack types[6]. Now we will discuss about the prevention and detection tools [3] for such attack types. Wassermann and Su propose Tautology Checker that uses static analysis to stop tautology attack. The weakness of this tool is that its scope is limited to tautology and cannot detect or prevent other types of attacks.

CANDID modifies web applications written in Java through a 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. It's natural and simple approaches turns out to be very powerful for detection of SQL injection attacks.

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

Two similar approaches by Nguyen-Tuong and Pietraszek modify a PHP interpreter to track precise per-character taint information. Livshits and Lam use static analysis techniques to detect vulnerabilities in software. Java Static Tainting uses information flow techniques to detect when tainted input has been used to make a SQLIA. The primary limitation of this approach is that it can detect only known patterns of SQLIAs and it can generate a relatively high amount of false positives because it uses a conservative analysis.

SQLPrevent is consists of an HTTP request interceptor[8]. The original data flow is modified when SQLPrevent is deployed into a web server. The HTTP requests are saved into the current thread-local storage. Then, SQL interceptor intercepts the SQL statements that are made by web application and pass them to the SQLIA detector module.

Consequently, HTTP request from thread local storage is fetched and examined to determine whether it contains an SQLIA. The malicious SQL statement would be prevented to be sent to database, if it is suspicious to SQLIA.

Xiang Fu and Kai Qian proposed the design of a static analysis framework, called SAFELI for identifying SQLIA vulnerabilities at compile time. SAFELI statically monitor the MSIL (Microsoft Symbolic intermediate language) byte code of an ASP .NET Web application, using symbolic execution. SAFELI can analyze the source code and will be able to identify delicate vulnerabilities that cannot be discovered by black-box vulnerability scanners. The main drawback of this technique is that this approach can discover the SQL injection attacks only on Microsoft based product.

Two approaches, SQL DOM and Safe Query Objects, use database queries encapsulation for trustable access to databases. They use a type-checked API which cause query building process is systematic. Consequently by API they apply coding best practices such as input filtering and strict user input type checking. The drawback of the approaches is that developer should learn new programming paradigm or query-development process.

Java Dynamic Tainting[7]and SecuriFly is another tool that was implemented for java. We discussed about the tools used for protecting against SQL Injection attack. Now we will discuss about another type of attack which is considered to be most dangerous than SQLI for web applications. We discussed about SQLI attack types, their prevention and detection techniques. Now further we will discuss about XSS attack and its types, which is considered to be more dangerous than SQLI attack.

Cross-Site Scripting Attack: (or XSS attack) is one of the most common application-layer web attacks. It commonly targets script embedded in a page which are executed on the client-side (in the user's web browser) rather than on the server-side[9] .XSS attack itself is a threat which is brought about by the internet security weaknesses of client-side scripting languages, with HTML and JavaScript (other being VBScript, ActiveX, or Flash) as the prime culprits for this exploit.

The concept of XSS is to manipulate client-side scripts of a web application to execute in the manner desired by the malicious user. Such a manipulation can embed a script in a page which can be executed every time the page is loaded, or whenever an associated event is performed. There are different types of XSS attacks.[1]

a. Non-Persistent or Reflected Attack (First Order XSS):

The most common attack, Considered less dangerous. Usually used in phishing attempts. Attack requires to persuade the victim to click on a prepared URL (Social Engineering).It can steal credential, deface a website, create a fake page or spam email, observe user request etc.

b. ***Persistent XSS Attack (or Stored or Second Order XSS Attack):***

It is similar to Non-persistent but even more effective. It is Stored in the attacked web server's database. Second-order XSS is much more damaging than first order XSS, for two reasons: (a) social engineering is not required (the attacker can directly supply the malicious input without tricking users into clicking on a URL), and (b) a single malicious script planted once into a database executes on the browsers of many victim users.

V. RELATED WORK

XSS has proven to be dangerous enough to consider and researches on Cross-site scripting attacks has been ongoing for a number of years now, and a large number of protection methods have been researched and tested, such as Modularized and configured solution to block XSS vulnerabilities based on service oriented architecture [4]**,**

Static Detection of Cross-Site Scripting Vulnerabilities [5] but these researches are helpful to avoid simple XSS attack; they are not able to detect Persistent XSS attack which is considered to be more dangerous than simple and first order attack. So our work area will be focus on persistent XSS attack.

In order to detect persistent XSS attack we will study about the attacker's scenario and understanding its behavior by simply creating an environment where attacker can intrudes some malicious code on to the webpage in order to steal the cookies, redirect the user on another website, can do port scanning and may also display or steal the content of the database. We also evaluate the attacker's behavior in order to prevent the persistent XSS attack.

VI. CONCLUSION

In this paper we analyzed various types of SQLI attacks. Then we investigated the strength and weaknesses of the tools used for protection and detection of SQL injection attack and its types. We conclude that some tools need improvement regarding their efficiency and performance. The tools and techniques used are limited to prevent and detect only particular type of SQLI attack not for group of attacks. Some technique must be developed which detect all kinds of SQLI attack. After that we discussed about XSS attack and types of XSS (first order and second order) attack. We found that second order XSS attack is more dangerous than first order attack.

In future work propose to create a mechanism for finding second orders XSS (persistent XSS attack) on the basis of escaping characters to counter the tautologies.

VII. REFERNCES

[1]. Adam Kie˙zun, Philip J. Guo, Michael D. Ernst and Karthick Jayaraman "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks", in: Proc. of the 31st Int Conf on Software Engineering ICSE '09, pp. 199-209.

[2]. Atefeh Tajpour and Mohammad JorJor zade Shooshtari Evaluation of SQL Injection Detection and Prevention Techniques. Liverpool, United Kingdom, July 28, 2010 to July 30, 2010, ISBN: 978-0-7695-4158-7,pp: 216-221.

[3]. Jayamsakthi Shanmugam,"A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture", 6th Annual IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), 11-13 July 2007, Melbourne, Australia.

[4]. Gary Wassermann and Zhendong Su, "Static Detection of Cross-Site Scripting Vulnerabilities" , ICSE May 2010, ISSN: 0270-5257, pp 171-180.

[5]. Divya Rishi Sahu and Deepak Singh Tomar, "Robust Defence against XSS through Context Free Grammar", International Journal of Computer Science and Technology, 2015/3, volume 6, issue 1, Version 2, pp 113-117

[6]. Prof.(Dr.) Sushila Madan and Ms. Supriya Madan "Security Standardsto Fortify Web Database Applications From Code Injection Attacks", 2010 International Confrence on Intelligent Systems, Modeling and Simulation.

[7]. Nenad Jovanovic, Christopher Kruegel, and Engin Kirda, "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities.", Technical University of Vienna, Secure Systems Lab, Proceedings of the 2006 IEEE Symposium on Security and Privacy(S&P06)1081-6011/06, 2006

[8]. Rahul Raj and D. K. Malhotra, "Forensic Investigation for Web Forgery through Java Script Obfuscation", International Journal of Computer Security & Source Code Analysis (IJCSSCA), 2015, Volume: 1, Issue: 1, pp. 05-08.

[9]. C M Mishra and R D Singh, "Web Defacement: Threat Analysis and Modeling", International Journal of Computer Security & Source Code Analysis (IJCSSCA), 2015, Volume: 1, Issue: 1, pp. 26-30.

[10]. Willium G. J. Halfond, Alessandro Orso and Panagiotis Manolios, Member IEEE Computer Society Published by the IEEE Computer society 0098-5589/08 2008.

