ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 1, pp: (46-52), Month: January - March 2016, Available at: www.researchpublish.com

A Survey to Detect and Prevent Web Attacks

[1]Mrunali P. Pathak, [2]Nida Kausar Khan, [3]Tejashree C. Tantak

Abstract: **Most of the cyber security techniques present today have many critical faults. This provides to be an openway for hackers and criminals. Using widespread techniques, hackers and criminals try to gain entry into one's system using these backdoors. However, gaining entry and retrieving information can be a tedious task for the hackers. Hence they use techniques such as SQL injection and Cross-Site Scripting (XSS) to obtain sensitive data such as password, account no etc. This paper analyses the source code of security patches of widely used web applications written in weak and strong typed languages. Results show that only a small subset of software fault types, affecting a restricted collection of statements, is related to security. To understand how these vulnerabilities are really exploited by hackers, this paper also presents an analysis of the source code of the scripts used to attack them. The outcomes of this study can be used to train software developers and code inspectors in the detection of such faults and are also the foundation for the research of realistic vulnerability and attack injectors that can be used to assess security mechanisms, such as intrusion detection systems, vulnerability scanners, and static code analysers.**

Keywords: **Intrusion detection, XSS, SQL injection, hackers, criminals, vulnerability, click-jacking, tab-nabbing.**

1. INTRODUCTION

Before discussing the concept of web-attacks, it is important to know how the web has evolved over time. The first generation web-based applications were very limited in their ability to cater to the needs of the user. To bridge this gap, the web applications evolved to give services like searching, posting and uploading.

CGI or Common Gateway Interface protocol provided a path to the user to interact with the web pages by submitting data into forms. CGI scripts would process the information at the back end and then submit it back to the end-user. This provided the very first attack vector for the web applications. Along- with CGI scripts, various frameworks like AJAX, Ruby on Rails, PHP, ASP.NET, and J2EE were developed to provide flexibility and manage workflow in web services.

Web Attack is simple an attack on a computer through the medium of a web based services like a website. The purposes of a web attack are varied and can be used for different reasons.

So what is a web attack made of? The basic web attack consists of a modified request that is designed to take disadvantage of the user's trust.

Web attack often compromises of various stages but in general, the basic 5 stages of a web attack are as follows:

1. **Entry Point:** The attacker visits a legitimate website and uploads a malware on the website. When the user visits this website, the malware uploaded on the website gets downloaded on the user's system without his/her knowledge.
2. **Distribution:** The malware will redirect the user to an exploit server depending on the user's OS.
3. **Exploitation:** Supported exploit packages will try to execute the vulnerabilities in the OS, browser, plugins etc.
4. **Infection:** The malware downloaded on the user's system will download a malicious payload on the system that will steal sensitive data from the user's system or session.
5. **Execution:** The malware will execute at the backend and trick the user into doing something without his knowledge.[6]

2. LITERATURE SURVEY

Jose Fonseca.*et.al.* The author have discussed the typical web faults are discussed considering various programming languages. In order to solve the problem of web security vulnerabilities, the authors have inspected the code patch to gather characteristics of code responsible for vulnerable behavior.[1]

S.Shalini.*et.al*, The authors have inspected the worms exploiting JavaScript XSS vulnerabilities. To counter this exploitation ,the authors have suggested a client side solution that uses a step by step approach to protect cross site scripting, without degrading the browsing experience.[2]

Lwin Khin Shar .*et.al*, The authors have presented the current approaches to mitigate this problem mainly focus on effective detection of XSS vulnerabilities in the programs or prevention of real time XSS attacks.To solve this problem, the authors have proposed method consists of two phases:

(1) XSSV detection and

(2) XSSV removal.

XSSV detection phase identifies potential XSSVs in the program source code using static analysis. XSSV removal phase first determines the context of each user input referenced in the identified potential XSSVs. It then secures the potential XSSVs by applying the appropriate escaping methods using an escaping library provided by ESAPI.[3]

William G.J. Halfond.*et.al*, This paper presents a new highly automated approach for protecting Web applications against SQL injection that has both conceptual and practical advantages over most existing techniques. The authors have stated that detecting.[4]

Kuldeep Kumar.*et.al*, In this paper, we proposed a method based on grammatical structure of an SQL statement and validation of the user input. This method uses combined static and dynamic analysis. The experiments show that the proposed method is effective and simple than other methods.[5]

3. PROBLEM DEFINITION

Design and implement a system such that it detects and prevents a potential Web Attacks.

Types of Web Attacks:

With the development of the Web and the websites on the web, the attackers found out many new attacks to harm the confidentiality of websites. Some of these attacks are non-existent but most of them are still working. The most common web attacks are as follows:

3.1 SQL Injection:

SQL injections allow the attackers to obtain unrestricted access to the databases running at the background of a web application. Due to this attack, the attacker can access financial as well as personal data of the user. SQL Injection attack takes place when the attacker changes the intended query by inserting new SQL keywords or operators in the query. There are various methods through which an SQL injection attack can take place:

- **Injection through cookies:** If a website uses cookies, the attacker can easily embed an attack in the cookies used by a website to simulate an attack.
- **Injection through user input:** The attacker can easily modify the user queries to insert SQL commands that fetch the data needed by the attacker.
- **Injection through server variables:** Server variables are collection of variables that contain HTTP headers, network headers and environmental variables. The attacker can directly place SQL attack query in the headers. The attack simulates when the query executes to log the server variable to the database.
- **Second-order injection:** This type of attack is quite different from the other attacks. This attack executes when the input i.e malicious query stored in the database is used.

We shall now discuss the various types of SQL injection techniques known along-with the intent of the attacker and the description of the attack.

Some SQL injection attacks are also done using magic code.

Example:

Username= admin

Password= ' or 1=1 #

3.1.1 Tautologies:

In this attack, the attacker intends to bypass authentication, identify inject able parameters and extract data. The attacker injects code into one of the conditional statements such that the condition always evaluates to true .The result of this attack depend on how the query is used in the application.

3.1.2 Illegal queries:

Through this attack, the attacker performs database finger-printing and data extraction. This attack forms the base for most other attacks as it leverages the vulnerability of an attack. The attacker tries to modify the query in such a way that it causes a syntax error, type conversion or a logical error. Syntax errors can be used to identify injectable parameters. Type conversion errors are be used to deduce the data types of certain columns or to extract data whereas logical errors often reveal the names of the tables and columns that caused the error.

3.1.3 Union Query:

The intent of the attack is to bypass authentication and extract data. The attacker exploits a vulnerable parameter to change the returned data set for the query.

He tricks the website to fetch data from a table which is not intended.

He does this by using the query of the following syntax:

UNION SELECT <rest of the injected query>

3.1.4 Piggy-Backed Queries:

The attacker uses this type of query to extract data, perform denial-of-service, execute remote commands and add/modify data. The attacker doesn't modify the query in this attack but adds new and distinct queries in the original query.

3.1.5 Stored Procedures

The intent of this attack is to perform privilege escalation, denial-of-service and execution of remote commands. As the attacker knows which database is functioning at the back-end, he can design SQL queries that can use the stored procedures including those that interact with the operating system. As the stored procedures are written in special scripting languages, the attacker can include arbitrary code that can escalate privileges or execute remote commands through the vulnerabilities included like buffer overflow. [6]

3.1.6 Inference:

The attacker identifies injectable parameters, extract the data and determines the database encoding through this type of attack. The query is modified to execute an action based on the answer to a true/false question. This attack is usually simulated on a website which has higher degree of security so that when the attack is successful, there will be no useful feedback to the via error reporting.

3.1.7 Alternate Encoding:

The attacker uses this attack to evade detection. This attack works in combination with other attacks where the injected text is modified to avoid detection. The combination with other attacks is necessary because the common evasion method is to scan the website for malicious content.

3.2 Cross-Site Scripting Attacks (XSS Attacks):

A malicious script can be embedded in the source code of the web page, the data that is submitted through forms available on the website or through any other means. This script can be designed for various purposes but are mainly used to steal

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 1, pp: (46-52), Month: January - March 2016, Available at: www.researchpublish.com

data from the user or the server. The script is a specially crafted hyperlink, usually written in JavaScript. The JavaScript has the ability to influence the behaviour of the browser.

The problems that can arise due to JavaScript are as follows:-

- Making changes to the local system, like changing the file system, copying data etc.
- Identifying keystrokes to steal confidential data like passwords
- Interact with the websites opened in the browser tabs without the user's knowledge.

The Cross-site Scripting attack takes the advantage of the vulnerabilities present in the website by implementing dynamic elements to compromise the security of the website.

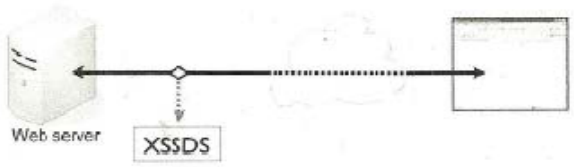

Figure 1: Overview of XSS Attack

Web browsers are designed in such a way that they do not let the website access documents loaded from other sites. Hence the attackers have to implement the XSS using different means. There are main three types of XSS attacks:

1. Non-persistent Attacks: This is the most common type of XSS attack. This attack targets the vulnerabilities present in the website when the data submitted by the client is processed at the server side to produce some results and the results are sent back to the client browser. The attacker tries to embed the malicious code in the results that are sent to the client browser. The attack is successful only when the malicious code is encoded into the Web page results, the results are sent to the client browser, where the code is not encoded using HTML special character coding and is not detected as inert text. The most common way of implementing an XSS attack is a link that involved badly-written URL.

2. Persistent XSS Attack: This attack works when the exploit is stored in the database at the server side. This exploit can be used to create malfunctioning webpages that will be displayed to other users later. The example of stored XSS attack is the forums and bulletins which allow the user to use HTML and XHTML to format their posts.

3. Dom-based XSS Attack: This is an unusual type of XSS attack wherein the logical errors in legitimate JavaScript and careless handling of client data results in XSS attack conditions.[1]

Cross-site scripting attack used the javascript code which actually creates an attack on the web site.

Example of Javascript code for attack: 1. <script>alert(0)</script>

2. <script> window.open("http://evilsite.com/cookie_stealer.php?cookie=" + document.cookie, "_blank"); </script>

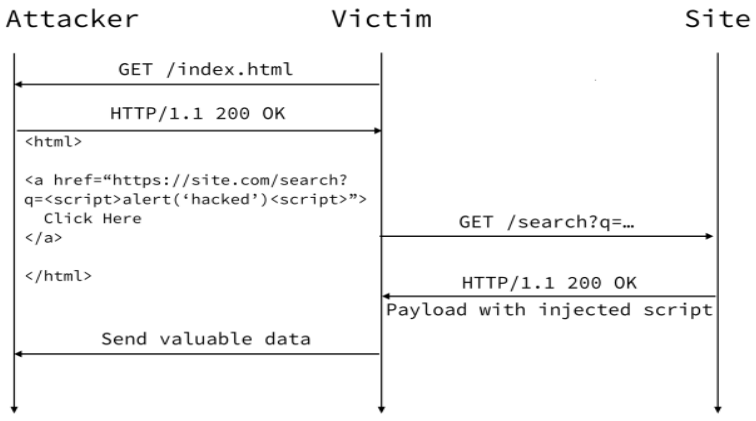

Figure 2: Non-Persistent XSS Attack Scenario

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 1, pp: (46-52), Month: January - March 2016, Available at: www.researchpublish.com

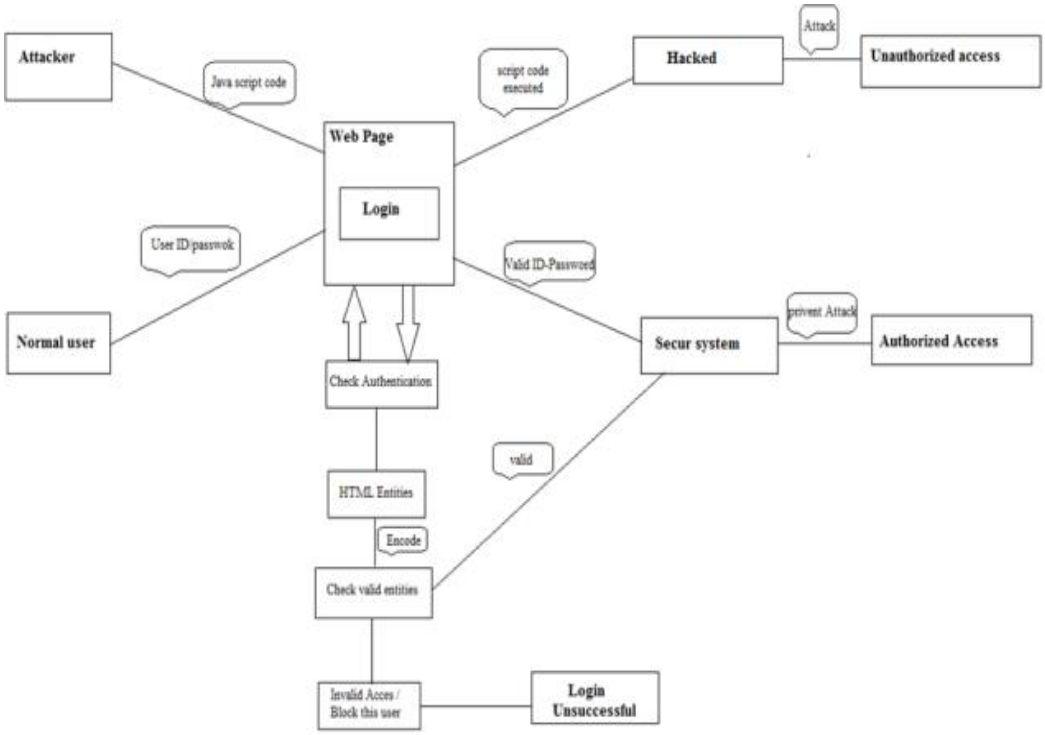

Figure 3: XSS Detection and Prevention Architecture

There are many challenges in defending the system against XSS attack which are listed as follows:

1. Encoding Schemes: If the URL is in an encoded form, it is not easily detected and hence the XSS attack can be easily implemented.

2. Anti-comment design flaws: The malicious code can be immediately filtered out automatically using the Auto-comment feature. This feature blocks the malicious code by commenting it.

3. Browser versions: Various browsers have various vulnerabilities. The modern day browser versions have an in-built capability of preventing the XSS attacks.

4. Tags: XSS codes that are embedded in some tags can execute automatically if sent by e-mail.

4. PREVENTION OF ATTACKS

How to prevent SQL injections?

There are many ways to prevent SQL injections and some of the techniques have been summarised as follows:

- Checking input types
- Input encoding
- Positive pattern matching
- Identification of all input sources

There are various approaches for detection of SQL injection and some of them are discussed as follows:

1. SQL-IDS Approach: This method analysis query specific allowing no computational overhead. This method does not provide any kind of negatives. It is a very efficient method.

2. Parse Tree Validation Approach: The SQL statement is compared with the parse tree of a certain statement at runtime. This method though very efficient, has some disadvantages which are introducing an additional overhead and lists only the inputs.

3. Manual Approach: Through this approach, the programmer makes use of defensive programming to introduce a filter to strain suspicious characters. Also, the creation of blacklists and whitelists helps to filter out malicious websites.

4. Using Stored Procedures: Stored procedures are functions that can be called to perform some operation on the database tables. This technique makes use of static and dynamic analysis. The Stored Procedure parser is used to identify SQL commands whereas SQL Checker is used at runtime to list the input.

5. SAFELI: This approach is identifies the SQL injection attack at compile time providing many advantages like White-box Static analysis and Hybrid Constraint Solver.

6. Database Security Testing: This approach has the following steps:-

- Detection of potential inputs for SQL injections.
- Automatic generation of test cases.
- Running test cases on a web application to find vulnerabilities.[7]

How to Prevent Cross-site scripting?

Cross-site scripting attacks are of two types reflected (non-persistent) and stored (persistent). To prevent this there are many precautions which user has to take.

1. Restrict Untrusted JavaScript

This method allow to run the JavaScript code only if it comes from a domain that the user explicitly trusts. Instaling a browser plug-in that implements domain whitelisting, such as NoScript for Firefox, is highly recommended. Internet Explorer users can achieve whitelisting through the configuration of Trusted and Restricted Security Zones.

2. Use Built-In Browser Protection:

Some browsers have begun to incorporate XSS protection inherently. For Example, as of version 8,Internet Explorer includes an XSS filter as well as a Smart Screen filter that uses reputation to protect against malicious websites. These extra security measures should be enable when available.[3]

3. Blacklisting vs. White Listing:

To help mitigate XSS attack, two basic techniques are used to sanitize data. Blacklisting uses a list of know bad data to block illegal content from being executed. Whitelisting uses list of know good data to allow only that content to be executed.

5. CONCLUSION

This project was designed specially to cater to the growing demands of the security domain and focuses mainly on the website attacks and their protection. We have handled various attack modules in this project viz SQLi, Cross-site Scripting attacks (XSS), Click-jack and tab-nabbing attacks. Through this project, we have aimed to build a system that can successfully detect these attacks and prevent them automatically. Some browsers already provide protection against these attacks but our system is designed to run universally. It is applicable in almost every field of work be it banking or e-commerce and can also be essential for people who work from home.

The future research will be considerate to constructAPIs to detect SQLi and XSS web attacks.

REFERENCES

[1] Jose Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira, Analysis of Field Data on Web Security Vulnerabilities"in IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 2, MARCH/APRIL 2014.

[2] S.SHALINI, S.USHA, Prevention of Cross-Site Scripting Attacks (XSS) On Web Applications in the Client Side.

[3] Lwin Khin Shar , Hee Beng Kuan Tan, Automated removal of cross site scripting vulnerabilities in web applications in 2011.

[4] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "A Classification of SQL Injection Attack and Countermeasures.

[5] Kuldeep Kumar, Dr. Debasish Jena and Ravi Kumar "A Novel Approach to detect SQL injection in web applications"

[6] F. Valeur, D. Mutz, and G. Vigna. A learning-based approach to the detection of sql attacks. LNCS, 3548:123–140, 2005.

[7] W.G. Halfond, J. Viegas, and A. Orso, "A Classification of SQLInjectionAttacks and Countermeasures," Proc. IEEE Int'l Symp. Secure Software Eng., Mar. 2006.

[8] http://www.slideshare.net/RespaPeter/types-of-sql-injection-attacks.

[9] https://blogs.sophos.com/2013/11/01/how-malware-works-anatomy-of-an-attack-in-five-stages-infographic/.

[10] Kanchana Natarajan, Sarala Subramani , "Generation of SQL-injection free secure algorithm to detectand prevent SQL-injection attacks".