International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(04), December 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354

Analysis of XSS and SQL Injection Attacks on Web Applications

Vruddhi Mehta[1] , Yash Gandhi [2] , Riya Muni[3] , Sakshi More[4]

[1 ,2,3,4]*Computer Engineering, SVKM's Shri Bhagubhai Mafatlal Polytechnic (India)*

ABSTRACT

In today's generation web applications are one of the most predominant platforms for information and service delivery over internet. The proliferation of attacks are rampantly increasing on the social networking applications, online transaction systems, e-commerce sites and such other valuable service oriented web applications. There are plenty of vulnerable websites on the Internet which should be se-cured against so many security threats. In this paper, we are demonstrating two web application penetration testing techniques that are XSS and SQL Injection and the remedial actions for the same.

Keywords:*SQL database, JavaScript, HTML*

I.INTRODUCTION

Web applications are used on a large scale world-wide, which handles sensitive personal data of users. With web application that maintains data ranging from as simple as telephone number to as important as bank account information, security is a prime point of concern. With hackers aimed to break-through this security using various attacks, we are fo-cusing on SQL injection attacks and XSS attacks. SQL injection attack is very common attack that ma-nipulates the data passing through web application to the database servers through web servers in such a way that it alters or reveals database contents. While Cross Site Scripting (XSS) attacks focuses more on view of the web application and tries to trick users that leads to security breach.

II. CROSS SITE SCRIPTING

Cross Site Scripting attacks are known as one of the main problems that web developers face in the web security field. XSS attacks consist in executing mali-cious scripts in the victim's browser using a prepared link or exploiting the website security so that the ma-licious code is delivered by the site itself. Exploiting this vulnerability allows to abuse the browser and steal data from it, including capturing the typed keys on the keyboard, showing non desired content and even stealing cookie's data (which can be used to supplant the client's session) and many other actions [1]

The consequence of an XSS attack is the same re-gardless of whether it is stored or reflected (or DOM Based). The difference is in how the payload arrives at the server. Do not be fooled into thinking that a "read only" or "brochure ware" site is not vulnerable to serious reflected XSS attacks. XSS can cause a va-riety of problems for the end user that range in sever-ity from an annoyance to complete account compro-mise. The most severe XSS attacks involve disclo-sure of the user's session cookie, allowing an at-tacker to hijack the user's session

and take over theaccount. Other damaging attacks include the disclo-sure of end user files, installation of Trojan horse programs, redirect the user to some other page or site, or modify presentation of content. An XSS vulnera-bility allowing an attacker to modify a press release or news item could affect a company's stock price or lessen consumer confidence. An XSS vulnerability on a pharmaceutical site could allow an attacker to modify dosage information resulting in an overdose.

III.IMPLEMENTATION

XSS attack works in two steps. In first and the main step attacker crafts an attack script and injects it through the application logic so that it resides over the web server. Once the same page, on which script is loaded, is revisited by some valid user the script run and fetches crucial information. Now this attack can be as simple as " " , which will insert a whole iframe in the comment section of a web application, luring victims to fill important data. Or script like," " can be used to steal cookie information of the user. [2]

1) *APPLLICATION CODE*:<input value="userInput">
2) *MALICIOUS STRING:* <script>…<\script>
3) *RESULTING CODE:* <inputvalue="<script>...<\script>">

Fig.3.A XSS Implementation

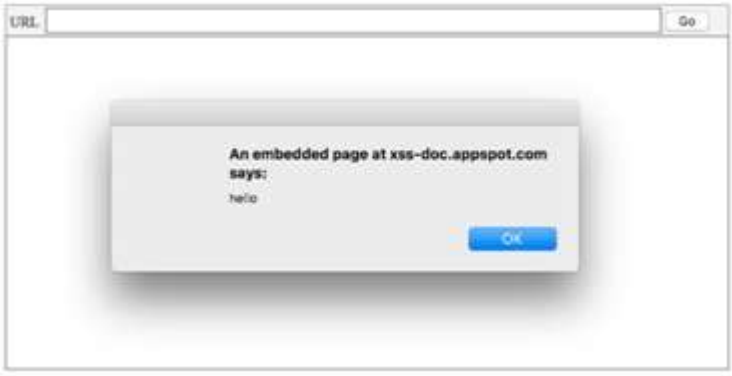

Fig.3.A XSS Result

IJARSE
ISSN (O) 2319 - 8354

Fig.3.B XSS Implementation

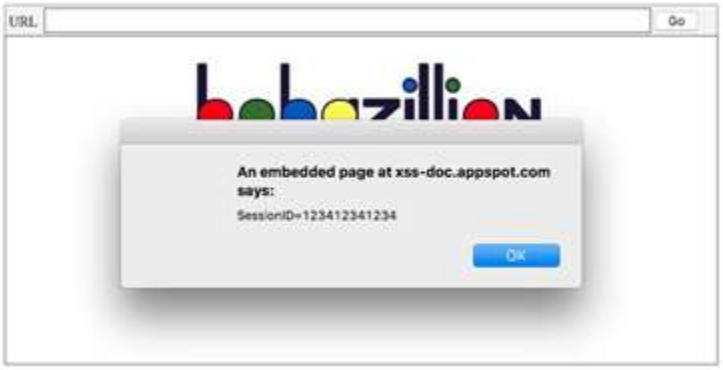

Fig.3.B XSS Result

IV.PREVENTION

A common technique for preventing XSS vulnerabil-ities is "escaping". The purpose of character and string escaping is to make sure that every part of a string is interpreted as a string primitive, not as a con-trol character or code.

For example, '<' is the HTML encoding for the '<' character. If you include: <script>alert('testing')</script> in the HTML of a page, the script will execute. But if you include: <script>alert('testing')</script>

In the HTML of a page, it will print out the text "<script>alert ('testing') </script>", but it will not ac-tually execute the script. By escaping the <script> tags, we prevented the script from executing. Tech-nically, what we did here is "encoding" not "escap-ing", but "escaping" conveys the basic concept (and we'll see later that in the case of JavaScript, "escap-ing" actually is the correct term).

The following can help minimize the chances that your website will contain XSS vulnerabilities:

- Using a template system with context-aware auto-escaping
- Manually escaping user input (if it’s not pos-sible to use a template system with context-aware auto-escaping)
- Understanding common browser behaviors that lead to XSS

- Learning the best practices for your technol-ogy. [3]

V. SQL INJECTION

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and be-come administrators of the database server. The se-verity of SQL Injection attacks is limited by the at-tacker's skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and so on. In general, consider SQL Injection a high impact severity. Since an SQL Injection vulnerability could pos-sibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities. [4]

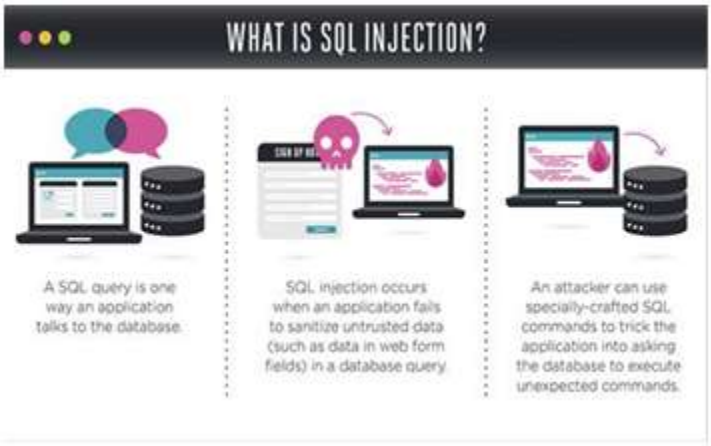

Fig.5 SQL Injection [9]

VI. DETAILED ANALYSIS OF SQL INJECTION

SQL is a programming language designed for man-aging data stored in an RDBMS (referred as control of web application's over its database server), there-fore SQL can be used to access, modify and delete data. Furthermore, in specific cases, an RDBMS could also run commands on the operating system from an SQL statement. , While considering follow-ing, it's easier to understand how lucrative a success-ful SQL Injection attack can be for an attacker.

(1)An attacker can use SQL Injection to bypass au-thentication or even impersonate specific users. (2) An SQL Injection vulnerability could allow the com-plete disclosure of data residing on a database server.

(3)Since web applications use SQL to alter data within a database, an attacker could use SQL Injec-tion to alter data stored in a database. Altering data affects data integrity and could cause repudiation is-sues, for instance, issues such as voiding transac-tions, altering balances and other records. (4) SQL is used to delete records from a database. An attacker could use an SQL Injection vulnerability to delete data from a database. Even if an appropriate backup strategy is employed, deletion of data could affect an application's availability until the database is re-stored. (5) Some database servers are configured (in-tentional or otherwise) to allow arbitrary execution of operating system commands on the database server. Given the right conditions, an attacker could use SQL Injection as the initial vector in an attack of an internal network that sits behind a firewall. [5]

Fig.6 SQL Working [10]

VII. IMPACTS OF SQL INJECTION

If your web application is vulnerable to SQL injection, a hacker is able to execute any malicious SQL query or command through the web application. This means he or she can retrieve all the data stored in the database such as customer information, credit card details, social security numbers and credential to ac-cess private areas of the portal, such as the adminis-trator portal. By exploiting an SQL injection, it is also possible to drop (delete) tables from the data-base. Therefore, with an SQL Injection the malicious user has full access to the database. Depending on your setup and the type of server software being used, by exploiting an SQL injection vulnerability some malicious users might also be able to write to a file or execute operating system commands. With such escalated privileges this might result into a total server compromise. Unfortunately, it is very difficult to determine the impact of an exploited SQL injec-tion. Most of the times, if the hackers are well trained Victim won't be able to detect the attack until your data is available to the public and your business rep-utation is going down the drain. [6]

VII. IMPLEMENTATION

8.1. Application code

PHP Code

$rs=mysql_query("select * from mst_user where login='$loginid' and pass='$pass'");

Java CodeString qs = "select * from EMPLOYEE where USER_NAME='" + userName + "' and PASS-WORD='" + password + "'";

8.2. Malicious code

1. ‘ or ‘1=1
2. ‘ or ‘‘=’
3. ‘ or true -- -

8.3.Result code

1. ‘ ’ or ‘ ‘=’ ’
2. ‘ ’ or ‘ 1=1’
3. ‘ ’ or true -- -

Fig. 8.A SQL Implementation

Fig. 8.B SQL Result[8]

IX. PREVENTION

Factors to prevent SQL injection attacks:

- *Constrain and sanitize input data:* -Checkfor known good data by validating for type, length, format, and range.

- *Use type-safe SQL parameters for data ac-cess:* -You can use these parameters with stored procedures or dynamically constructed SQL command strings. Parameter collections such as SqlParameterCollection provide type checking and length validation. If you use a parameters collection, input is treated as a lit-eral value, and SQL Server does not treat it as executable code. An additional benefit of using a parameters collection is that you can enforce type and length checks. Values out-side of the range trigger an exception. This is a good example of defense in depth.
- *Use an account that has restricted permis-sions in the database:* - Ideally, you shouldonly grant execute permissions to selected stored procedures in the database and provide no direct table access.
- *Avoid disclosing database error information:* - In the event of database errors, make sureyou do not disclose detailed error messages to the user [7].

Code to avoid malicious data input into the database and protecting our database from untrusted source (i.e. hackers)
PreparedStatement stmt = connection.prepareState-ment("SELECT * FROM users WHERE userid=? AND password=?");
stmt.setString(1, userid); stmt.setString(2, password); ResultSet rs = stmt.executeQuery();
This code is not vulnerable to SQL Injection because it correctly uses parameterized queries. By utilizing Java's PreparedStatement class, bind variables (i.e. the question marks) and the corresponding setString methods, SQL Injection can be easily prevented. Due to this the attacker cannot inject malicious code and in input since it does not allow to bypass such varia-bles eg: ‘ , @ ? # into the database.

X. CONCLUSION

The World Wide Web is growing day in and out, people at large are getting connected with one or more applications and the applications are exchanging information as a when required for reducing the end user's connection time. The applications are re-questing information which the user has filled in some other application to gain access to his confidential information like user ID and password. The ap-plication-to-application inter-process communication is actually a loophole to end user's information. Majority attacks to web applications today are mainly carried out through input manipulation in or-der to cause unintended actions of these applications. Most of the attackers inject vulnerable scripts from the input boxes which collect data from the user's credit card information box (XSS) or the username and password box (SQLI). These text boxes are highly vulnerable to the websites and need to be tested thoroughly for all the possible attacks. Thus giving end user correct message in case any misleading information is entered into the required text boxes. As here, we attacked a website using XSS and SQL Injection to gain access over the websites with-out knowing the actual username and passwords by playing around with the scripts. We also introduced some techniques to prevent these attacks.

XI. ACKNOWLEDGEMENT

We express our gratitude towards our guide Mr.Manish Solankifor carrying us through the entireproject “Analysis of XSS and SQL Injection Attacks onWeb Applications. “

REFERENCES

[1] "An analysis of XSS, CSRF and SQL injection in colombian software and web site development - IEEE Conference Publication", Ieeexplore.ieee.org, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7520140/

[2] Detection of SQL injection and XSS attacks in three tier web applications - IEEE Conference Publication", Ieeexplore.ieee.org, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7860069/.

[3] [Online]. Available: https://www.google.com/about/appsecurity/learn- ing/xss/. [Accessed: 12- Dec- 2017].

[4] "SQL Injection - OWASP", Owasp.org, 2017. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection.

[5] Online]. Available: http://www.acunetix.com/websitesecurity/sql-injec-tion/.

[6] 2017. [Online]. Available: https://www.netsparker.com/web-vulnerability-scan-ner/vulnerability-security-checks-index/sql-injection/. [Accessed: 12- Dec- 2017].

[8] 2017. [Online]. Available: https://www.google.com/about/appsecurity/learn-ing/xss/.

[9] "SQL Injection Cheat Sheet & Tutorial: Vulnerabilities & How to Prevent SQL Injection Attacks", Veracode, 2017. [Online]. Available: https://www.veracode.com/security/sql-injection.

[10] Hackertarget.com, 2017. [Online]. Available: https://hackertarget.com/xss-simple-tutorial.png.