SQLi and XSS Attack Introduction and Prevention Technique

Harshad Gaikwad
Dept. Computer Engg.
Srttc Khamshet,
Pune.

Bhavesh B. Shah
Dept.Computer Engg.
Srttc Khamshet,
Pune.

Priyanka Chatte
Dept. Computer Engg.
Srttc Khamshet,
Pune.

ABSTRACT

Nowadays, web applications are common around the world. every major company/organization have a web application presence. Max of these organizations use web applications to provide various services to clients. Some of these web applications employ database driven content. The back-end database often contains confidential and sensitive information such Password, credit card number, financial data, medical data, email details. Typically the web user/client supplies information, such as a username and password and web server receive user request and interact with the back-end database and returned relevant data to the Front-end.

Web Applications penetration testing and security has become progressively most important these days. A lot numbers of malicious attacks are being deployed on the web application. Due to dramatic increase in Web applications usage, Web application get vulnerable to variety of threats. Most of these malicious attacks are targeted towards the web application layer and waf firewall alone cannot prevent these kinds of attacks. The reason behind success of these attacks is the ignorance of application developers while coding the web applications and the predefined vulnerabilities in the existing technologies. Web application attacks are the latest trend and hackers are trying to hack/exploit the web application using different techniques. Various types of solutions are available as open source and in market. But the selection of suitable solution for the security of the organizational systems is a major issue. Some Attack Prevention Technique protect web applications from attacks they sit in front of web applications monitors activity, and block malicious traffic.

Keywords

SQL injection attack, SQL query, XSS (cross site scripting), Web application, Payload, filters.

1. INTRODUCTION

Web applications are wildly spread to provide services and became usefull communication channel between service provider and client. this web applications mostly uses java script for validation and verification instead of web application firewalls which cause serious security problems.

XSS can steal the user session cookies and use current session for modification and SQLi attacker can directly access database.

This two attacks are listed as top 2 attacks by OWASP community.

2. SQLI MECHANISAM

2.1 Injection through user input

In the type of injection the attacker injects SQL injection commands by providing suitably crafted user input. A web application can read user's input in several ways based on the environment in which the application is deployed.

2.2 Injections through cookies (Cookie injection point)

Cookies are the small stored files that containing state information generated by Web applications and stored on the client machine. When a client returns to the Web application the cookie is used to be restore the client information. Since the client has control over the storage of cookie, a malicious client could tamper with the cookie's content. And then if Web application uses the cookie content to build SQL queries, an attacker could easily submit an attack by embedding it in the cookie.

2.3 Injections through the server variables

Server variables are collections of variables that contain HTTP, Network headers and environmental variables. Web applications used these server variables in a variety of ways like logging usage. If these servers logged to a database without sanitization, this could create SQL injection vulnerability because attacker can forge the values that are placed in HTTP and network headers. They can exploit this vulnerability by placing an SQL injection directly into the headers. And when the query to log the server variable is issued to the database, the attack in the forged header is triggered automatically.

2.4 Second order injection

Not easy to inject this attack.In second order injection, attacker/hacker need malicious inputs in to a system or database to indirectly trigger an SQL injection when that input is used at a later time. The attack takes place when the malicious input reaches to the database.

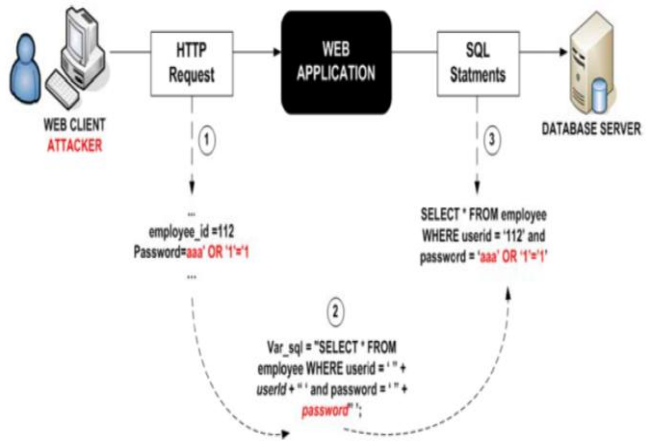

WEB CLIENT
ATTACKER
HTTP Request
WEB APPLICATION
SQL Statments
DATABASE SERVER
1
...
employee_id =112
Password=aaa' OR '1'='1
...
2
Var_sql = "SELECT * FROM employee WHERE userid = ' " + userId + " ' and password = ' " + password" ';
3
SELECT * FROM employee WHERE userid = '112' and password = 'aaa' OR '1'='1'

Fig 1: SQL injection Flow Diagram.

3. CLASSIFICATION OF SQLI
3.1 Tautology based

In the tautology attack the attacker tries to use a conditional query statement to be evaluated always true. Attacker uses true and false conditions with WHERE clause to inject and turn the condition into a tautology which is always true. The simplest form of tautology.

Example :

$sql_query = "select * from Report where userid = 'user11' and password = 'anything' or 'a'='a' ";

The result would be all the data in accounts table because the condition of the WHERE clause is always true

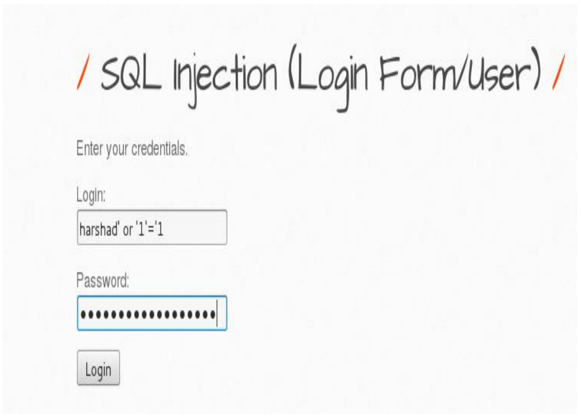


Fig 1: Tautology Based SQL injection.

3.2 Error Based

In this Injection the attacker inputs the different types of vectors like single quote to break the application to see the errors and create a payload as per the error to do sql injection

www.vuln-web.com/photo.php?id=111'--

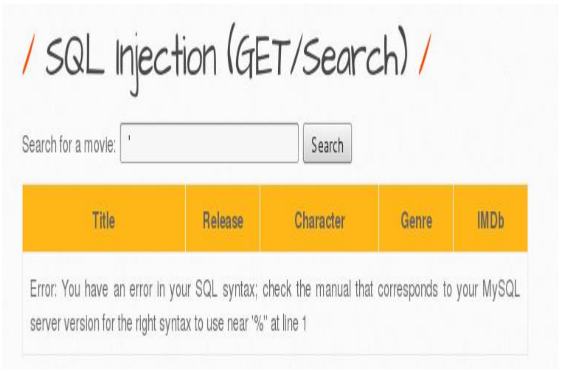
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' LIMIT 0,1' at line 1

www.vuln-web.com/photo.php?id=111'#

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1

www.vuln-web.com/photo.php?id=111'/*

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '/*' LIMIT 0,1' at line 1



Title	Release	Character	Genre	IMDb

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1

Fig 2: Error Based SQL injection.

3.3 Union Based Queries

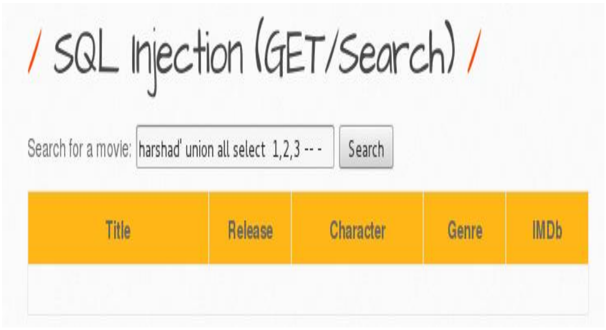
In this type of queries unauthorized query is attached with the authorized query by using UNION clause.

$sql_query = Select * from report where id="32 "union based injection here" ;

Example :

http://vul-site.com/report.php?id=32" union all select 1,2,3--+

The result of the first query in the example given above is null and the second one returns all the data in Report table so the union of these two queries is the Report table.


/ SQL Injection (GET/Search) /
Search for a movie:
harshad' union all select 1,2,3 -- -
Search

Title	Release	Character	Genre	IMDb

Fig 3: Union Based SQL injection.

3.4 Blind Injection

This is little difficult type of attack for attacker. During the development process sometime the developer hides some error details which help the attacker to compromise with database. In this situation the attacker face the generic page provided by developer in place of an error message

Example :

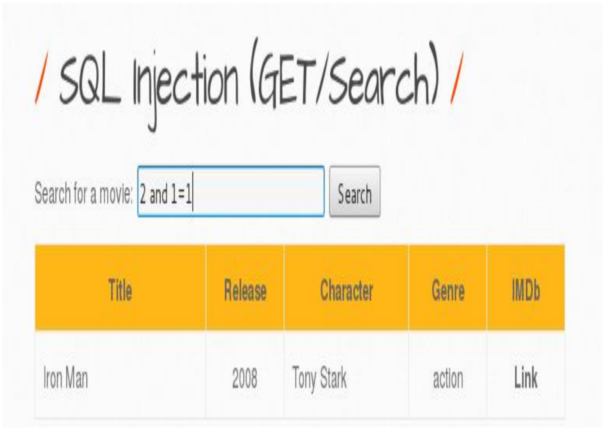
$sql_query=Select * from Report where id=111' and false%32

www.vuln-site.com/photo.php?id=111' and true%32

Normal Page returned.

www.vuln-site.com/photo.php?id=111' and false%32

Page didn't Load As normally it do as the query didn't returned anything.



Title	Release	Character	Genre	IMDb
Iron Man	2008	Tony Stark	action	Link

Fig 4: Blind SQL Injection.

Blind SQL Injection is used when there is No Output and No Error from the web application, that means we can't inject the Union based injection in which we use to get the output nor we can Inject the XPATH or Sub Query Injection which use to get the output in form of Error. While doing a Blind injection we make Queries from the database and ask if we are right or wrong.

3.5 Time based injection

In the Timing attack the attacker gathers information about the response time of the database. This technique is used by executing the if-then statement which results the long running query or time delay statement depending upon the logic injected in database and if the injection is true then the "WAITFOR" keyword which is along with the branches delays the database response for a specific time.

Example :

$sql_query= select * from report where id=41' wait for delay '00:00;10'

www.vuln-site.com/photo.php?id=41' wait for delay '00:00:10'--+

Delay in page loading

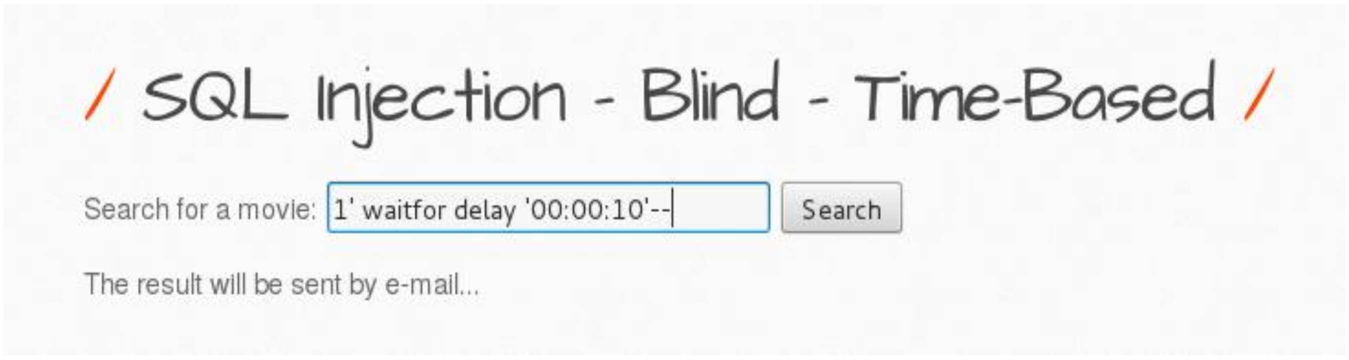

/ SQL Injection - Blind - Time-Based /
Search for a movie:
1' waitfor delay '00:00:10'--
Search
The result will be sent by e-mail...

Fig 5: Time Based SQL injection.

4. PREVENTION OF SQLI

To prevent sql injection there are some core techniques. Which helps to prevent sqli injection attack. Listed Below.

4.1 Defensive coding

Developers have approached a range of code based development practices to counter SQLI. These techniques are generally based on proper input filtering, potentially harmful character and rigorous type checking of inputs.

4.2 Manual defensive coding practices

Based on the security reports such as OWSAP's SQL cheat sheet

4.3 Parameterized queries or stored procedures

The attacker take advantage of dynamic SQL by replacing the original queries and create some parameterized query in database. These attacks force to developer for first define the SQL code structure before including parameters in query.

Because parameters are bound to the defined SQL structure, there after it is not possible to inject additional SQL code

4.4 Escaping

If dynamic queries cannot be avoided, escaping all user-supplied parameters is the best option. Then the developer should identify the all input sources to define the parameter that need escaping, follow database-specific escaping procedures, and use standard defining libraries instead of the custom escaping methods.

4.5 Data type validation

After following the steps for the parameterized query and escaping the developer must properly validate the input data type. The developer must define the input data type is string or numeric or any other type and input data given by user is incorrect then it could easily reject.

4.6 White list filtering

Some of the special character which is normally used during injection. so the developer should characterize such special character as the black list filtering.

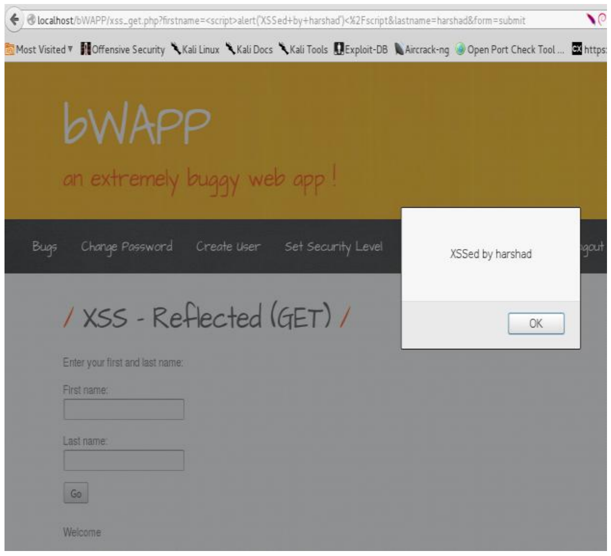
The filtering approach is suitable for the well structured data. Such as email address, dates, etc. and developer should keep a list of legitimate data patterns and accept only matching input data

5. XSS ATTACK (Cross-site scripting)

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page

An attacker inserts a XSS payloads <script> alert (document.cookie) </script> in the input Field and send it to the server, when server get that request server process it as a legal request and send back the result to the attacker.



```
localhost/bWAPP/xss_get.php?firstname=<script>alert('XSSed+by+harshad')<%2Fscript&lastname=harshad&form=submit
```

XSSed by harshad

Fig 6: XSS Attack.

6. CLASSIFICATION OF XSS

6.1 Stored XSS

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a comment field, message forum, visitor log etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-I XSS.The most dangerous XSS type. security risk is high.

6.2 Reflected XSS

Reflected attacks are those where the injected payload is reflected from the web server, such as in form of error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web site. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

6.3 Dom based XSS

DOM Based XSS (or as it is called in some texts, "type-0 XSS") is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

7. PREVENTION OF XSS

7.1 XSS Filter to Block Most XSS Vectors

There is a simple rule that should be follow to prevent XSS attack: Encode every data that is given by a user. If data is not given by a user but supplied via the GET parameter, encode these data too. Even a POST form can contain XSS vectors. So, every time you are going to use a variable value on the website, try Sanitization for XSS.

These are the main data that must be properly sanitized before being used on your website.

- The URL
- HTTP referrer objects
- GET parameters from a form
- POST parameters from a form
- Ajax Request

First Thing to do, encode all html tags like <, >, ' and ". This should be the first step of your XSS filter. Encoding should be :

&	=>	&amp;
<	=>	&lt;
>	=>	&gt;
"	=>	&quot;
'	=>	&#x27;
/	=>	&#x2F;

For this, you can use the htmlspecialchars() function in PHP. It encodes all HTML tags and special characters.

$Some_input = htmlspecialchars($input, ENT_QUOTES);

If the $Some_input was= "><script>alert(XSS)</script>

This function would convert it into &quot;&gt;&lt;script&gt;prompt("XSS")&lt;/script&gt;

A vector may use HTML characters, so you should also filter these. Add this rule :

$Some_input = preg_replace('/(&#*w+)[x00-x20]+;/u', '$1;', $data);

$Some_data = preg_replace('/(&#x*[0-9A-F]+);*/iu', '$1;', $input);

There are many places where input does not need script tags. An attacker can inject a few event functions to execute scripts. And there are many ways by which an attacker can bypass this filter. So, we need to think about all possibilities and add a few other things to make the filter stronger. And not only JavaScript, you also need to escape from cascading style sheets and XML data to prevent XSS.

7.2 Open Source Libraries for Preventing XSS Attacks

7.2.1 PHP Anti-XSS

This is a great PHP library that can help developers to add an extra layer of protection from cross site scripting vulnerabilities. It automatically detects the different types of encoding of the data that must be filtered. Very easy to apply on web applications.

7.2.2 HTML Purifier

This is a standard HTML filtering library written in PHP. It Block all malicious payloads from the input Fields and protects the web applications from XSS attacks. It is also available as a plug-in for most PHP frameworks.

7.2.3 XSS HTML Filter

This is another great XSS filter for Java web applications. It is a simple single-class utility that can be used to properly filter/sanitize user input against cross site scripting and malicious HTML code injection.

Some of this libraries are also built to prevent different attacks like command injections. Which by default filters the & (and), | (pipe) and ; (semicolon) symbols.

8. ACKNOWLEDGMENTS

We would like to take this opportunity to express our profound gratitude and deep regard to our project guide Prof. Bhavesh B. Shah, for his guidance, valuable feedback and for constant encouragement for the project. Working under him was extremely knowledgeable experience for us.

9. CONCLUSION AND FUTUREWORK

Paper defines various types of XSS and SQL injection attacks. then investigation of XSS and SQLi detection and prevention technique. after that examples of XSS and SQLi attacks.

In our future work ,listed prevention technique have some drawbacks also so Strong technique must implemented to stop XSS and SQLi. automated separate tool can be implemented to stop XSS and SQLi attack.

10. REFERENCES

[1] Chaitali Khairnar, “Detection and Automatic Prevention against SQL Injection Attack and XSS Attacks perform on web application,” Maharashtra india, vol. 5, issue 11,november 2015. .

[2] Kuldeep Kumar, Dr. Debasish Jena and Ravi Kumar.”A Novel Appraoch to detect SQL injection injection in Web application”. 2013,InstaSafe Technologies Pvt. Ltd, Bangalore-560076.

[3] Atefeh Tajpour, Suhaimi Ibrahim, Maslin Masrom, “SQL Injection Detection and Prevention Techniques” International Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011

[4] Punam Thopate, Purva Bamm, Apeksha Kamble, Snehal Kunjir, Prof S.M.Chawre"Cross Site Scripting Attack Detection & Prevention System".International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)Volume 3 Issue 11, November 2.

[5] Cross-site Scripting (XSS): https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[6] Open web Application security project, XSS(cross site scripting).prevention cheat sheet,2011; http://www.owasp.org/index.php/Xss_(Cross_site_scripting))_preventation_cheat_Sheet