# A Review of Web Application using Cross Site Scripting (XSS)

**Ruchika**

*Abstract*— **In today's era, web applications are rapidly used by the people around the world. People are contingent on vivid web applications for daily needs such as online banking, shopping, gaming, booking tickets, using webmail and the list is endless. These applications are developed using various languages like PHP, C#, ASP.NET, Python, PERL using scripts like HTML and JavaScript. Some of these applications are vulnerable to attacks. Attackers develop tools and techniques to exploit user data due to flaws in security. All this defines the need of efficient web application security. The most common attacks on web applications are by XSS which are commonly used techniques by attackers for cyber-crimes**

.

## I. INTRODUCTION

Web application security deals with the protection of web services, web applications, websites and all other benefits of web. The web applications use two types of scripts: Server side script and Client side script. The former scripts (ASP, PHP etc.) include hard stuff i.e., storing and drawing the information while the latter ones, HTML, JavaScript etc., deal with the display of information.

During the past year, attacks on web applications have grown rapidly[1]. The two significant techniques of attack are Structured Query Language (SQL) Injection and Cross Site Scripting (XSS), apart from PHP injection, Java injection, Memory corruption, Path disclosure and Denial of Service.

Nowadays, a significant part of the population use online services to file their income tax, fill passport forms, banking services, etc. using one of these applications. This information would always be on stake if there isn't high security. They invade these applications and get access of confidential data, cause severe loss to the client and can also damage the server. The web application security is needed to prevent raid by attackers.

## II. WEB APPLICATION SECURITY

Web Application Security falls under the category of Information security. The design interfaces, which the user uses to interact with backend databases, must be secure to perform the tasks over web applications[2]. In some severe cases attacker can attain system level permissions and direct access to database which can harm the server and execute the commands according to their choice. This accounts for the need for web application security. Generally, it can be divided into two parts: Declarative security and Program security.

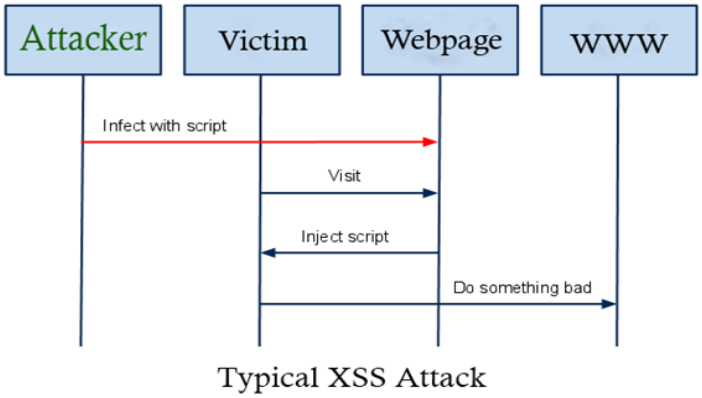
### Declarative Security

It comprises of mechanisms used in an application, and shown in declarative syntax in a deployment descriptor (DD). A DD is an .xml file that shows application security structure including access controls and authorization requirements[7]. The security information is placed into annotations by declarative security syntax. The metadata includes:

- Description of the assembly i.e., identity, name and type.
- Details of types.
- Attributes.

### Program Security

It is done using programs hence user has full access and authentication control. Consequently, it makes all the components portable as there is involvement of server related components. Also, there is no need to create configuration files. The programs can be written according to the need. For better authentication and user friendly experience, a customized program can be made. The programs give a flexible experience in the development of web applications. Though this communication is slow, it is a safe process. All these aspects help in building dynamic web pages whose content varies according to the arguments passed[11].




Typical XSS Attack

Figuure1.1

- Scripting**:** Web Browsers can execute commands

- Embedded in HTML page



ISSN: 2278 – 7798

-Supports different languages (JavaScript, VBScript, ActiveX, etc.)

-Most prominent: JavaScript

- "Cross-Site" means: Foreign script sent via server to client

- Attacker makes Web-Server deliver malicious script code

-Malicious script is executed in Client's Web Browser.

- Example when a malicious user injects a script in a legitimate shopping site URL which in turn redirects a user to a fake but identical page. The malicious page would run a script to capture the cookie of the user browsing the shopping site and that cookie gets sent to the malicious user who can now hijack the legitimate user's session.

## III. CROSS SITE SCRIPTING (XSS/CSS)

It is a very usual application layer hacking techniques used since 1990s. It chiefly targets the social networking websites such as Facebook, Twitter, and Orkut.

- Most common application layer web attack
- Also known as XSS attack
- It targets scripts embedded in a page which are to be executed on client side rather than server side
- To manipulate client side scripts in a page which can be executed in the manner desired by the malicious user
- The hacker infects a legitimate web page with his malicious client side script
- Example when a malicious user injects a script in a legitimate shopping site URL which in turn redirects a user to a fake but identical page. The malicious page would run a script to capture the cookie of the user browsing the shopping site and that cookie gets sent to the malicious user who can now hijack the legitimate user's session.

## IV. XSS ATTACK TYPES

There are three distinct types of XSS attacks: non-persistent, persistent, and DOM-based [8].

**Non-persistent** cross-site scripting vulnerability is the most common type. The attack code is not persistently stored, but, instead, it is immediately reflected back to the user. For instance, consider a search form that includes the search query into the page with the results, but without filtering the query for scripting code[4]. This vulnerability can be exploited, for example, by sending to the victim an email with a special crafted link pointing to the search form and containing a malicious JavaScript code. By tricking the victim into clicking this link, the search form is submitted with the JavaScript code as a query string and the attack script is immediately sent back to the victim, as part of the web page with the result. As another example, consider the case of user who accesses the popular trusted.com web site to perform sensitive operations (e.g., on-line banking).

**Persistent** type stores malicious code persistently in a resource (in a database, file system, or other location) managed by the server and later displayed to users without being encoded using HTML entities. For instance, consider an online message board, where users can post messages and others can access them later. Let us assume further that the application does not remove script contents from posted messages. In this case, the attacker can craft a message similar to the next example[6].

This message contains the malicious JavaScript code that the online message board stores in its database. A visiting user who reads the message retrieves the scripting code as part of the message. The user's browser then executes the script, which, in turn sends the user's sensitive information from his site to the attacker's site.

**DOM-based** cross-site scripting attacks are performed by modifying the DOM "environment" in the client side instead of sending any malicious code to server. So the server doesn't get any scope to verify the payload.

It is a four stage process:

1. Attacker inserts client side script into vulnerable web server or web pages.
2. A user (victim) visits this server.
3. The code enters the victim's browser.
4. The code executes with web server privileges

## V. MITIGATION

1. To mitigate XSS attack, you must prefer escaping of string input. The script should include the location of the unwanted strings to be placed in a HTML document.
2. You must employ additional security measures when running a website functioning on cookie based authentication.
3. Use plug-ins which could block untrusted websites.
4. Blocking the scripts would be beneficial if you have knowledge about them. Many defensive technologies are available which guarantees minimum XSS.

## VI. CONCLUSIONS

Web application security is the demand of the hour. Online business is most affected by web application vulnerabilities. The main problem is that most of the web applications don't have testing of desktop software. Therefore there is always a growing concern for security. There must be a way to deal

with the unexpected input from a browser, unhandled error messages, unchecked database call. You can use content management system like Joomla! to construct a website with dynamic content. There can also be a way to prevent injections which is to avoid characters which have a distinct meaning in SQL. Single quote (‘) can be replaced by two single quotes (‘‘) to form an accurate SQL string literal. You must assign limited permissions to the database and extend these when the need arises.

You must prefer PHP over any other language as PHP is moving towards object oriented architecture and there is use of PDO classes which give privilege to make prepared statements to prevent SQL Injections.

In order to secure your web applications you must be aware about the above said attacks and should be prepared for any kind of new attacks. Use web application vulnerability evaluation tools in order to check the current security level of your web application or website. Educate developers, security professionals, testers regarding the risks and steps to mitigate the attacks. You must be sure how you pass the data and don’t expose your logic. A single loop hole can lead to the destruction of your whole database. You must always have a backup and keep it safe.

.

REFERENCES


[1] Kim-Kwang Raymond Choo (2011) The cyber threat landscape: challenges and future research directions computers & security, Volume 30, Issue 8, Pages 719–731

[2] Kindy, D.A. and Pathan, A.-S.K., “A Survey on SQL Injection: Vulnerabilities, Attacks, and Prevention Techniques” in Proc. 15th IEEE Symposium on Consumer Electronics, Singapore, 2011, pp. 468-471.

[3] Klaus Julisch (2013) Understanding and overcoming cyber security anti-patterns, Computer Networks, Volume 57, Issue 10, pp 2206–2211

[4] Robin Dyer, (2013) "External reactive detection v. internal proactive prevention: The holistic approach to integrate change", Journal of Financial Crime, Vol. 20 Iss: 3, pp.287-292

[5] Tajpour, A., Masrom, M., Heydari, M.Z., and Ibrahim, S., “SQL injection detection and prevention tools assessment,” in Proc. 3rd IEEE International Conference on Computer Science and Information Technology, China 2010, pp. 518-522.

[6] E. Athanasopoulos, V. Pappas, and E. Markatos. Code-Injection Attacks in Browsers Supporting Policies. In Proceedings of the 2nd Workshop on Web 2.0 Security & Privacy (W2SP), Oakland, CA, May 2009.

[7] A. Barth, J. Caballero, and D. Song. Secure Content Sniffing for Web Browsers or How to Stop Papers from Reviewing Themselves. In Proceedings of the 30th IEEE Symposium on Security & Privacy, Oak-land, CA, May 2009.

[8] A. Barth, J. Weinberger, and D. Song. Cross-Origin JavaScript Capability Leaks: Detection, Exploitation, and Defense. In Proceedings of the 18th USENIX Security Symposium, Montreal, Quebec, August 2009.

[9] S. W. Boyd and A. D. Keromytis. SQLrand: Pre-venting SQL Injection Attacks. In Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference, pages 292–302, 2004.

[10] S. Chen, D. Ross, and Y.-M. Wang. An Analysis of Browser Domain-Isolation Bugs and a Light-Weight Transparent Defense Mechanism. In Pro-ceedings of the 14th ACM conference on Computer and Communications Security (CCS), pages 2–11, New York, NY, USA, 2007. ACM.

[11] S. Designer. Return-to-libc attack. Bugtraq, Aug, 1997


**I received** B.Tech. degree in computer science from Deen Bandhu Chotu Ram University of Science and Technology, India in 2012 and pursuing my M.Tech degree from Deen Bandhu Chotu Ram University of Science and Technology, India. My area of interest is to find the efficient method to protect web applications using XSS.



ISSN: 2278 – 7798