*International Journal of Scope of Computer Research*
*Vol. 5, No. 4, pp. 034-037, April 2016*
*http://www.wrpjournals.com/IJSCR*

**Research Article**

# A Survey on XSS - The Top Web Application Security Vulnerability

**Shivangi Sharma and *Dr. Sheetal Kalra**

Department of Computer Science, Guru Nanak Dev University, Regional Campus, Jalandhar Regional Campus, Jalandhar, India

*Received Article 26th March, 2016; Published Article 27th April 2016*

Web application dependence is increasing very rapidly day by day for social communications, financial transaction and various other purposes. Unfortunately, security risks present due to weaknesses present in source code in web applications allows malicious user's to exploit various security vulnerabilities and become the reason of their failure. Due to vulnerabilities present in web applications, they can be continuously targeted by malicious user and thus performing attacks. Among these attacks, Cross-Site Scripting (CSS or XSS) attack is a common type of attacks. This study has identified Cross Site Scripting as the top web application security vulnerability. This paper surveys the previous work on Cross Site Scripting detection and prevention using Dynamic Cookies Rewriting and the technique for PHP using ARDILLA tool.

**Key words:** Cross site scripting, Web applications, Web proxy, Cookie.

***Corresponding author: Shivangi Sharma***
*Department of Computer Science, Guru Nanak Dev University, Regional Campus, Jalandhar Regional Campus, Jalandhar, India.*

## INTRODUCTION

XSS has become the most common threat in web applications. Web applications are the software applications that can be run and accessed on network with the use of web browsers. They can be categorized as static web application and dynamic web application. Static web applications are those that display the information to the user and dynamic web applications accepts input from the user and does actions based on the input (Dr. Jayamsakthi Shanmugam and Dr. Ponnavaikko, 2008). A significant amount of web applications are vulnerable. The reason behind vulnerabilities is the weaknesses present in source codes and improper input handling. The weaknesses present in source code include programming language weakness, improper input validations or ignorance of security guidelines by developers who repeat the same type of programming mistakes in their code (Mukesh Kumar Gupta *et al*., 2014). XSS vulnerability is the result of lack of web application in sanitizing/validating user inputs. Using these un-sanitized inputs, malicious users inject malicious scripts in web pages of web applications. XSS attacks possess various severe security risks which includes identity theft, access to restricted or confidential information, altering the functionality of browser, redirecting user to another websites etc. XSS provides platform for other type of attacks like Cross Site Request Forgery, Session Hijacking etc. (Vikas *et al*., 2013). XSS attack includes the insertion of a piece of script on the client side. The code can be written in any language among which JavaScript is mostly used. Generally, personal firewall and antivirus software on the client are invalid to XSS attacks. XSS attackers may use encoding, encryption and other technologies to escape or avoid the server-side filtering (Ding Lan *et al*., 2013). Web proxy can effectively reduce the traditional XSS attacks that aim to steal user's information. XSS flaw occurs when an application takes untrusted data and sends it to a web browser without proper validation or escaping. It allows attackers to execute scripts in victim's browser thus allowing attackers to hijack user sessions or redirect user to malicious sites. XSS is one of the most common application layer attack technique used by attacker.

## BACKGROUND

### A. XSS Attack Types

There is no standardized classification of the XSS attacks. The two well-known types of XSS attacks: Non-persistent and Persistent.

#### i) Non-persistent (or reflected) XSS

Also known as first-order XSS or Type 1 XSS vulnerabilities occur when a user is directed to a web application that has XSS vulnerability, by the malicious user. Once the user gets to the website or application the malicious user's attack is executed. In reflected XSS attacks, the injected code doesn't reside on the web server (Vikas *et al*., 2013). The attack is crafted by a series of URL parameters that are sent via a URL. The malicious links are sent to victims using email, instant messages, blogs or forums or any other possible methods. When user clicks on this link, the injected code goes to the web server of attacker, which sends the attack back to the victim's browser. Browser will execute this code as it comes from a trusted server thus performing malicious work like stealing the confidential information of victims, hijack user's account via cookie etc.

**ii) Persistent (or stored) XSS**

They are also known as second-order XSS or Type 2 XSS. In stored or persistent XSS attacks the injected code is permanently stored on the target servers. The storage of method can involve a database, or a wiki, or blog. Stored XSS attacks are generally performed on web applications that take input from user in the form of text and store in the database of web applications (Vikas *et al.*, 2013). Examples of such applications are blogs, forums etc. These attacks are more dangerous than reflected as reflected attacks are dynamic in nature whereas stored attack can just be set once.

**B. Cookies**

A cookie (also called Internet cookie, HTTP cookie, Browser cookie, web cookie) is a small piece of data which is sent from a website that user is browsing and is stored in user's browser. Whenever user loads the website next time, the browser sends the cookie back to the server to notify the website about the user's previous activity. They provide stateful communications over the HTTP. Cookies are given ID tags. These are broadly used to store the session IDs or personal sensitive information. In general, cookies can be categorized into two types:

- **Session cookies:** They are also known as an in-memory cookie or transient cookie. They are used temporarily while user navigates the websites; are discarded when the user closes the browser; do not have an expiration date assigned to them.
- **Persistent cookies:** They expire at a specific date or after a specific length of time; are stored on the disk.
- Security vulnerabilities may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access (with the user's credentials) to the website to which the cookie belongs. There are two different versions of cookie specifications in use (Rattipong Putthacharoen and Pratheep Bunyatnoparat, 2011):
- **Version 0 cookie (Net Scape cookie):** It is the most widely used version. The cookies are identified by the combination of the following attributes: name, domain, and path. An arbitrary string can be used by the web server as the value of the name attribute.
- **Version 1 cookie (RFC 2965):** It is an extended version of Net Scape cookie. In addition, it adds an ability to identify the cookie by the port attribute.

The web server must always specify the value of the name attribute and the cookie version in the cookie, and the browser must return the same values. Almost all modern browsers do not support the version 1 cookie except Opera browser, so the version 1 cookie is not widely used by web developers.

**C. Web Proxy**

The web proxy is an intermediate that fulfills transactions on behalf of clients. Many organizations allow the users only to access the internet via the web proxy. The web proxy can be a separate device or a part of a firewall. It must sit between the clients and the web servers, and acts as both the client to the web server and the web server to the client, as shown in Figure1. All web connections from the clients are intercepted at the web proxy, and then the web proxy will initiate new web connections to the web servers on behalf of the clients. Proxy servers also help improve security by filtering out some web content and malicious software. It helps improve web performance by storing a copy of frequently used web pages.

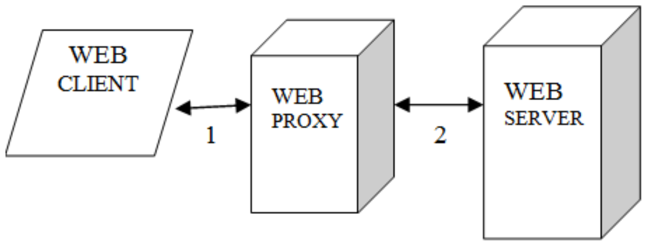

**Figure 1. Web proxy**

**1: Connection 1:** Client opens connection; Proxy intercepts it and impersonates Server.
**2: Connection 2:** Proxy opens connection to Server; forward server's responses back to client through connection1

## CROSS SITE SCRIPTING (XSS) DETECTION AND PREVENTION

1. "*Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique".(Rattipong Putthacharoen, Pratheep Bunyatnoparat) (2011)*

(Rattipong Putthacharoen and Pratheep Bunyatnoparat, 2011) This approach present how the cookies can be significantly protected from the XSS attacks. The approach is implemented in the web proxy without requiring any change on both web browser and web server. This technique is called "Dynamic Cookie Rewriting". With this technique, web proxy will automatically rewrite the value of the name attribute in the cookie with the randomized value before it sends cookie to browser. Browser will keep the randomized value in its database instead of original value. At web server end, the return cookie from browser will be rewritten again to its original form at the web proxy before being forwarded to the web browser. Now, the browser's database does not store original values of cookies so when there is a XSS attack, the cookies that are stolen from browser's database cannot be used later to impersonate the users. To implement this approach in the web proxy and take it into the action in the live environment, at least following three challenges have to be considered:

- Compatibility – This approach changes the values of the cookies in HTTP header, so it have to be made sure that this will not break the HTTP and be able to work well and transparently with all websites on the internet. Although almost all websites on the internet use only the version 0 cookies but this approach follows both version 0 and version 1 cookie specifications in order to be sure that it is able to manage properly all cookies sent by all websites.
- Performance –Taking the web proxy into the action onto a network definitely adds more latency and increases the response time. So with the web proxy in place, instead of degrading the system performance, it helps to multiply overall performance, improve user experience and also reduce internet bandwidth usage.

- Single point of failure - Web proxy rewrites the cookies, and keeps the original values of the cookies in its database. So when it fails, those original values will be gone. Fortunately, there are several technologies in place which are able to detect and bypass the web proxy once it fails such as PAC file, WCCP Protocol, Policy Based Routing (PBR) and so on. With one of those technologies, even the web proxy fails, the clients are still able to reach the web servers.

2. *"Automatic Creation of SQL Injection and Cross-site Scripting (XSS) Attacks (Ardilla)" (Adam Kiezun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst)(2009)*

(Adam Kiezun *et al.*, 2009) This approach presents a technique and an automated tool for finding security vulnerabilities in web applications. The technique works on unmodified existing code, creates concrete inputs that expose vulnerabilities, operates before software is brought into action, has no overhead for the released software, and analyzes application internals to discover vulnerable code. As an implementation of this technique, an automated tool Ardilla is created for creating SQLI and XSS attacks in PHP/MySQL applications. Ardilla is a white-box testing tool, i.e., it requires the source code of the application. It is designed for testing PHP applications. Security vulnerabilities that Ardilla identifies can be fixed before the software reaches the users because Ardilla creates concrete attacks that exploit the vulnerability. It is based on input generation, taint propagation, and input mutation to find variants of an execution that exploit vulnerability. The user of Ardilla needs to specify the type of attack (SQLI, first-order XSS, or second-order XSS), the PHP program to analyze, and the initial database state. The outputs of Ardillaare attack vectors.

**Ardilla**

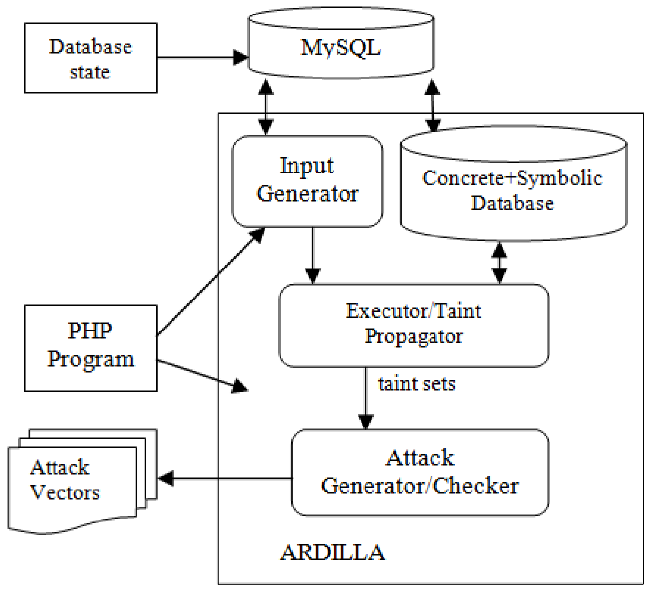

**Figure 2. The architecture of ARDILLA.**

Limitations that exist with this approach:

- Developer availability and learning is required.
- Source code adjustment is needed.
- If the original developer left the project it is very difficult to patch the vulnerabilities.
- It is tested on PHP based applications.

3. *"Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey"(Mukesh Kumar Gupta, Department of Computer Engineering, Malviya National Institute of Technology, Jaipur, India) (2014)*

(Mukesh Kumar Gupta and Govil, Girdhari Sing, 2014) This paper proposes a classification of software security approaches used to develop secure software in various phase of software development life cycle. It also presents a survey of static analysis based approaches to detect SQL Injection and cross-site scripting vulnerabilities in source code of web applications. The aim of these approaches is to identify the weaknesses in source code before their exploitation in actual environment. The vulnerability detection approach is classified into static analysis, dynamic analysis and hybrid analysis. In static analysis approach, it analyzes the source code without running the application. Various static analysis techniques include lexical analysis, type inference, dataflow analysis, constraint analysis, symbolic execution etc. Dynamic analysis approach requires executable code to detect the vulnerabilities. It includes fault injection testing technique (introduces faults to test behavior of system) and dynamic taint based approach (tainted data is inspected to check validity of input before their use). Limitation of this approach is only vulnerabilities present in the execution paths are detected, unable to detect vulnerabilities in parts of code that were not executed, results produced are not generalized for future executions. These two approaches are applied in coding or testing phase of SDLC. Hybrid analysis approach combines static and dynamic analysis. This paper would help researchers to note down future direction for securing legacy web applications in early phases of software development life cycle.

## Conclusion

It is found that Cross-Site Scripting (XSS) attacks are most powerful and easiest attack methods on the Web Applications. This paper presents a survey of techniques for defending against XSS exploits. The promising strategies such as Dynamic Cookies Rewriting, ARDILLA have developed which are rising. Dynamic Cookie Rewriting aims to disarm the attackers from impersonating the users. ARDILLA is an automated tool that can effectively and accurately find the most damaging type of input-based Web application attack: stored (second-order) XSS for PHP based web applications. This study shows the strengths and weaknesses of current approaches against cross site scripting. The main motivation of this work is to summarize the recent approaches in this field of research, identify major issues and challenges and to encourage further research in this field.

## REFERENCES

Adam Kiezun, Philip J. Guo, Karthick Jayaraman, Michael, D. 2009. Ernst, "Automatic Creation of SQL Injection and Cross-site Scripting (XSS) Attacks (Ardilla)" *IEEE* .

Ding Lan and Wu Shu Ting, Ye Xing and Zhang Wei, 2013. "Analysis and Prevention for Cross-site Scripting Attack Based on Encoding" *IEEE*.

Dr. Jayamsakthi Shanmugam and Dr. M. Ponnavaikko, 2008. "Cross Site Scripting-Latest developments and solutions: A survey" *Int. J. Open Problems Compt. Math.,* Vol. 1, No. 2, September 2008.

Mukesh Kumar Gupta, M.C. and Govil, Girdhari Singh, 2014. "Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey" IEEE International Conference on Recent Advances and Innovations in Engineering, (ICRAIE-2014), May 09-11, Jaipur, India.

Rattipong Putthacharoen and Pratheep Bunyatnoparat, 2011. "Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique"Feb. 13~16, ICACT2011. Method for Detecting Cross-Site Scripting Attacks.

Vikas K. Malviya, Saket Saurav and Atul Gupta, 2013. "On Security Issues in Web Applications through Cross Site Scripting (XSS)" 20th Asia-Pacific Software Engineering Conference.

*******