A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques

Shaimaa Khalifa Mahmoud
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University, Cairo, Egypt
shaimaa_khalifa.cs@yahoo.com

Marco Alfonse
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University, Cairo, Egypt
marco@fcis.asu.edu.eg

Mohamed Ismail Roushdy
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University, Cairo, Egypt
mroushdy@cis.asu.edu.eg

Abdel-Badeeh M. Salem
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University, Cairo, Egypt
abmsalem@yahoo.com

***Abstract*—Now the web applications are highly useful and powerful for usage in most fields such as finance, e-commerce, healthcare and more, so it must be well secured. The web applications may contain vulnerabilities, which are exploited by attackers to steal the user's credential. The Cross Site Scripting (XSS) attack is a critical vulnerability that affects on the web applications security. XSS attack is an injection of malicious script code into the web application by the attacker in the client-side within user's browser or in the server-side within the database, this malicious script is written in JavaScript code and injected within untrusted input data on the web application. This study discusses the XSS attack, its taxonomy, and its incidence. In addition, the paper presents the XSS mechanisms used to detect and prevent the XSS attacks.**

Keywords— Cross Site Scripting (XSS), web security, JavaScript code, injection code, Malicious JavaScript, XSS vulnerability, web application security, Stored attack, Reflected attack, DOM-base attack.

I. INTRODUCTION

The web application is an application program runs on the server "server-side" and is accessed through the web browser "client-side" to share the information and services (i.e. online shopping websites, social networking websites and internet banking… etc.)[1, 2]. The web application consists of the server side (encoded into Personal Home Pages (PHP), Active Server Pages (ASP) and Java Server Pages (JSP)) and client side (encoded into JavaScript, Visual Basic Script (VBScript), Hyper Text Markup Language (HTML), ActiveX). The client-side includes static web pages with scripting languages such as JavaScript executed within the browser by HyperText Transfer Protocol (HTTP) request [2]. The web applications have many vulnerabilities that may be exploited by attackers to steal the sensitive information. The organizations spend high cost to be secured from these vulnerabilities. According to the statistics of Open Web Application Security Project top10 - 2017 (OWASP), the XSS attacks are one of the top web application vulnerabilities in recent years [3, 4]. The XSS attack classification is divided into Reflected (Non-persistent), stored (persistent) and DOM-based XSS attacks [5,6].

The rest of this paper is organized as follows; section II presents the process, taxonomy, the statistical threat vulnerability of XSS attacks, and the incident occurred in the last 5 years, section III contains a survey on the detection and prevention techniques of XSS attacks in the last 7 years and presents a comparison between them. Finally, section IV contains the conclusions and future work.

II. CROSS SITE SCRIPTING ATTACKS

The XSS attack is a malicious JavaScript code executed in the victim's browser to steal the user's credentials such as cookies, credit card numbers, and passwords… etc. [6]. Fig. 1 shows a high-level view of a typical XSS attack. The XSS attacks are divided into three types namely; Reflected XSS attack, Stored XSS attack, and DOM-based XSS attack[7].

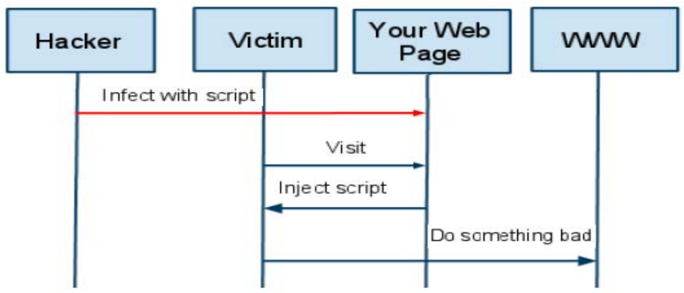


Figure 1. A high-level viewing of typical XSS attack [7]

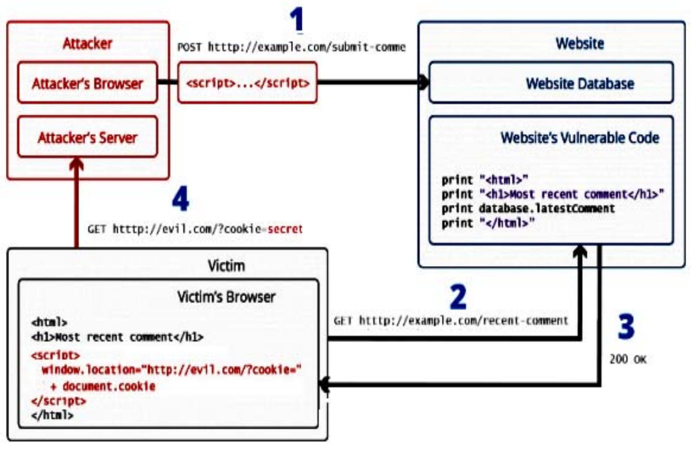
The XSS attack process, the taxonomy of XSS attack, the statistics of XSS attacks serious vulnerability and the incident XSS attack are discussed in the following subsections.

A. XSS attack process:

The hackers inject a malicious script code into the web page and when the user visits this website, the malicious script exploits the credentials of the user by hijacking session ID, password, credit card number or cookies [3, 4, 6, 8]. The XSS attack process is illustrated in Fig. 2. The process consists of the following steps:

978-1-5386-0821-0/17/$31.00 ©2017 IEEE

1. When the attacker finds a vulnerability in a web page, he/she injects a malicious script code (JavaScript code) into this vulnerable web application to steal the sensitive credentials from the victim's web browser.
2. The victim visits the infected web page.
3. The website sends the requested page with the malicious code as part of the HTML body.
4. Once the malicious script code is executed inside the victim's browser, the attackers steal the personal information from the victim's cookies.


1
POST http://example.com/submit-comme
Attacker
Attacker's Browser
Attacker's Server
<script>...</script>
Website
Website Database
Website's Vulnerable Code
4
GET http://evil.com/?cookie=secret
Victim
Victim's Browser
2
GET http://example.com/recent-comment
3
200 OK

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
print "</html>"
```

```
<html>
<h1>Most recent comment</h1>
<script>
 window.location="http://evil.com/?cookie="
   + document.cookie
</script>
</html>
```

Figure 2. The XSS attack process [8]

B. The taxonomy of XSS attacks:

XSS is classified into two main classes, server-side vulnerability and client-side vulnerability [5, 6, 9].

1. Server-side vulnerability:

The server-side vulnerabilities occur more frequently than the client-side vulnerability since most of the web applications are generated by dynamically scripts and contain the user input and responses to user's requests. The server-side vulnerability has two types; Reflected XSS attack and Stored XSS attack.

a. **Reflected XSS attack (Non-persistent attack)**

The purpose of this attack is to steal the session cookie of the user. This type of XSS attack requires a more interaction between the victim and the attacker [3, 5]. The steps of the Reflected XSS attack are illustrated in Fig. 3. The steps are as follows:

1. The attacker sends a link containing malicious script code to the user via email or any similar web page.
2. when the user clicks on this link, the malicious code is sent to the server without being detected by the web application
3. The server sends the HTTP response to the user with the malicious script code.
4. The attacker's domain receives the user's cookies after executing the script contained in the response.
5. The attacker can store these cookies for future use.

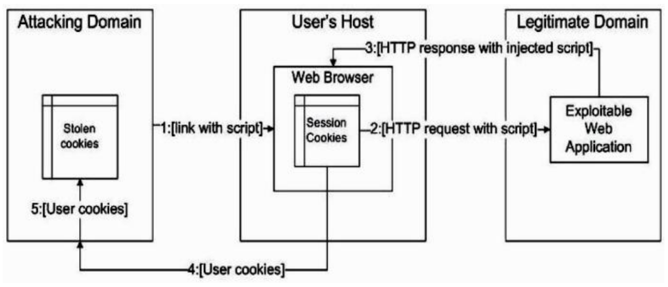


Figure 3. The Reflected XSS attack [5]

b. **Stored XSS attack (persistent attack)**

This kind of attack occurs frequently in the social networks and other similar applications. The steps of the Stored XSS vulnerability are illustrated in Fig. 4. These steps are as follows [3,5]:

1. The attackers insert a malicious script code on a web application that has a vulnerability.
2. When the user sends HTTP request, the web page content which includes the malicious code is accessed.
3. The malicious script is sent to the user by the HTTP response.
4. The script is executed in the web browser and sends the session cookies to the attackers.
5. These stolen cookies are stored on the attacker's domain.

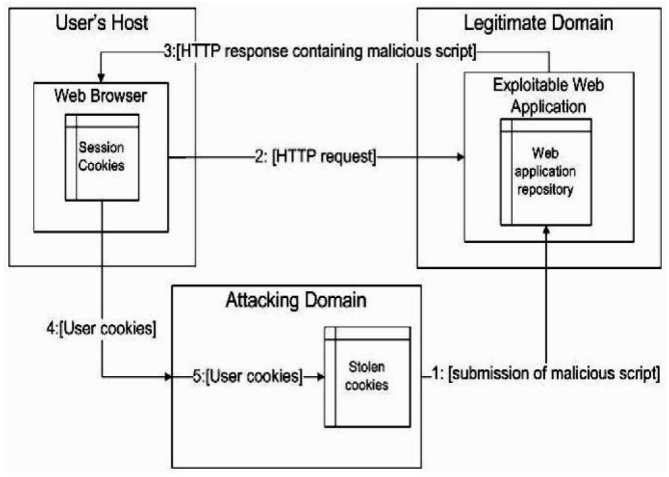

User's Host
Legitimate Domain
3:[HTTP response containing malicious script]
Web Browser
Exploitable Web Application
Session Cookies
2: [HTTP request]
Web application repository
Attacking Domain
4:[User cookies]
5:[User cookies]
Stolen cookies
1: [submission of malicious script]

Figure 4. The Stored XSS attack [5]

2. Client-side vulnerability:

This kind of vulnerability is an injection attack that executes a malicious code within the user's browser [4]. The XSS attack. The DOM-based XSS attack is considered as client-side vulnerability. The **DOM-based XSS attack** occurs in Document Object Model (DOM) rather than in the HTML code. In Fig. 5, the steps of the DOM-based XSS attack are illustrated [5, 6]. These steps are as follows:

1. The user receives a link contains malicious script from the attacker via email or bulletin board or through a similar webpage.

2. The user uses this malicious link. The request and the response do not contain malicious script.
3. The malicious script code is executed at the client and sends the user's cookies to the attacker's domain.
4. The attacker can store these stolen cookies for future use.

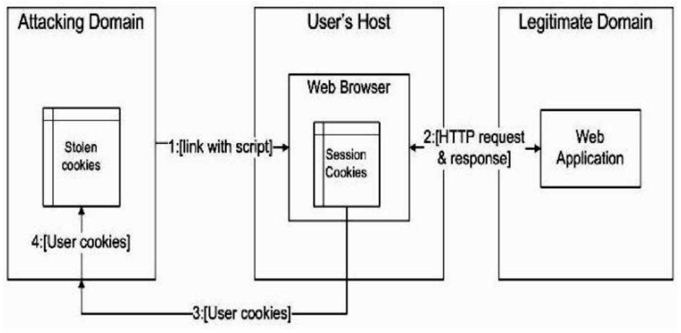


Figure 5. The DOM-based XSS attack [5]

C. *The statistics of XSS vulnerabilities:*

According to OWASP Top 10-2017, XSS is the third vulnerability in the list of the web application security vulnerabilities [4]. While it falls in the fourth level in the serious "high severity and critical" vulnerabilities in 2017 in the Dynamic Application Security Testing (DAST) [10]. Fig. 6 shows the XSS attacks serious vulnerabilities by DAST.

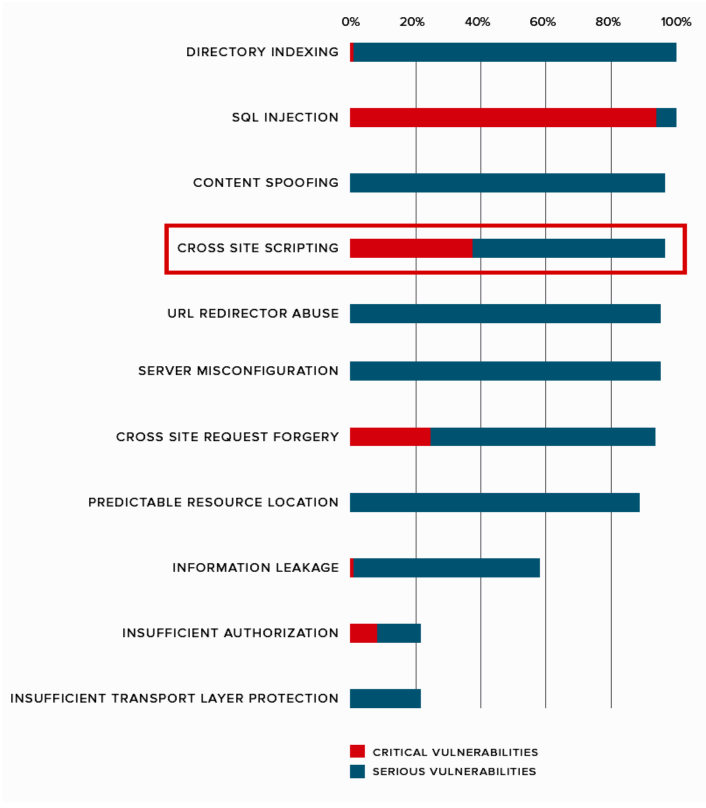


Figure 6. XSS attacks serious vulnerabilities by DAST [10]

D. *The incident of XSS attacks:*

The XSS attacks that are occurred during the last five years are illustrated in table I. These attacks target many global websites such as Facebook, Twitter, Yahoo, PayPal, Google… etc.

TABLE I: XSS ATTACK (2011-2016)

Target	Incident details	Consequence
Magento e-commerce platform, Jan 2016 [11]	Discovered XSS Bug in Magento puts millions of e-commerce merchants at risk of takeover, the XSS bug is present in virtually all versions of Magento Community Edition and Enterprise Edition prior to 1.9.2.3 and 1.14.2.3, It allows attackers to embed malicious JavaScript code inside customer registration forms. Magento executes the scripts in the context of the administrator account, making it possible to completely take over the server running the e-commerce platform.	Account hijacking
Google Cloud, May 2016 [12]	Discovered XSS vulnerability in Google by Patrik Fehrenbach.	Disinformation
eBay's website, December 2015 [13]	A Reflected XSS attack is discovered on eBay's website that gives the attackers the potential to steal millions of user credentials	Account hijacking
eBay's website, April 2015 [14]	Jaanus Kääp found XSS bug on eBay, the XSS allow an attacker over eBay's internal messaging system by catching and tweaking a request. This is a Stored XSS type.	Disinformation
UK Parliament Web Site, March 2014 [6]	XSS discovered in the search engine on the UK Parliament website. This search engine requires the user to write a code for retrieving the images, videos and each request for passwords.	Disinformation
Video Web Site, April 2014 [6]	This vulnerable relied on a Stored XSS vulnerability in video Web site and hijacked each of user's browsers for flooding the site by intolerable congestion.	DDoS attacks
Yahoo Web Site, January 2013 [6]	Using DOM-based XSS vulnerability to exploit the Yahoo email user's account by a hacker named "Shahin Ramezany". The Yahoo email users had been either compromised with receiving the spams from any other Yahoo users or clicking on the malicious links.	Account hijacking
PayPal Web Site, March 2012 [6]	Discover XSS attack in the login of PayPal web pages. The attacker injects malicious script code via URL and hijacking the login web page and steal the sensitive credentials such as password and user-ID.	Account takeover
Hotmail Web Site, June 2011 [6]	Discover XSS vulnerability in Hotmail email web services, it can steal keystrokes, cookies from the sensitive information.	Session hijacking
Facebook XSS Worm[15]	This exploits the Facebook with self-propagating spam worm by sending a spammed/malicious message to the user to visit a malicious site.	Worm

III. Detection and prevention of XSS attacks

The researchers have implemented the XSS prevention and detection techniques by using the open source code guidelines from different resources such as OWASP Secure Coding Practices", "SANS TOP 25 Most Dangerous Software Errors" and "CERT Coding Standards"… etc.[16]. In testing, they use two methods; automated and manual. The automated testing is performed with a commercial scanner tools such as Acunetix WVS/ Acunetix, AppScan/IBM, Rapid7 AppSpider, NetSparker/ Mavituna Security, Web Inspect/HP, WebApp 360/Trip Wire… etc.[16,17]. The manual testing is performed by penetration testing which uses proxy tools such as Zed Attack Proxy (ZAP)/OWASP, Fiddler or Burp Suite [4,16].

Shahriar and Zulkernine [18], developed S^2XS^2 tool to detect XSS attack automatically from the server-side by using the concept of "policy generation" to validate the user input data and the ''boundary injection'' to encapsulate the dynamic-generated content based on HTML tags and JavaScript code injection. This tool is encoded into java and uses the Jericho HTML parser and Rhino parser to parse the source of pages and identify the features of JavaScript code. The authors evaluate this approach on four real worlds JSP programs.

Shahriar and Zulkernine [19], developed a prototype tool to detect the XSS attacks from the server-side with injected comment statements that consist of the randomly generated tokens and some of the benign JavaScript codes as a server-side filter. This tool is encoded into java. This approach uses the Jerricho and Rhino parser to parse the source file and JavaScript code respectively. It is evaluated with three randomly open sources JSP programs in the real world.

Barhoom and Kohail [20], developed a server-side solution technique to detect and prevent the Stored-XSS attack for pages written with PHP and HTML programming language. This solution detects any injected script that breaks the rules of the schema by using the GET/POST input from HTTP request and any dynamic printing from the database by storing the Extensible Markup Language Schema Definition (XSD) as an input field and a dynamic part of the web application page to detect the untrusted user input that is stored in the schema.

Shar and Kuan Tan[21], developed SaferXSS tool to detect and remove the XSS vulnerability from the server. This tool is composed of three levels; program analyzer, XSS detector, and XSS removal. It uses the pattern matching techniques and static analysis with taint-based analysis techniques for the detection of the XSS attack. The XSS attack removal removes the escape untrusted data by using the OWASP Enterprise Security APIs (ESAPIs) and OWASP prevention rules through the sanitization method.

Gundy and Chen[22], created Noncespaces tool to prevent XSS attacks by using the Instruction Set Randomization (ISR) techniques to differentiate between benign and malicious contents for thwart the XSS vulnerabilities exploitation.

Huajie Xu et al. [23], proposed a defensive model based on user browsing behavior and website logic structure of the client event to protect the website and clients against XSS attacks by the analysis of the logical structures of the websites and user browsing behavior then loading the analyzed website behavior source code into Extensible Markup Language (XML) file to parse the logic structure and record the navigation, if it is illegal then the model display a warning message or blocks the website.

Parameshwaran et al. [24], developed DEXTERJS tool to detect and prevent the DOM-based XSS vulnerability on the web application by using the taint tracking mechanism. The tool is evaluated by the Alexa top 1000 sites which contain 820 distinct zero-day DOM-XSS.

Gupta and Gupta[1], proposed an automated detection system that is used to automatically discover the malicious JavaScript code injection vulnerabilities on the web application within the client-side JavaScript libraries after performing some modifications on the browser. This system is evaluated with the PHP web application BlogIT by three key agents; XSS Attack Vector Injector, HTML Crawler, and Script Locator.

Gupta and Gupta [25], developed a framework to enhance a defensive methodology of XSS in the cloud platforms by scanning the HTTP requests and compare its URL links embedded with the HTTP response script content by using the ISR techniques to thwart the exploitation of the XSS vulnerability. This model is deployed in the Virtual Cloud Server (VCS). The framework enhances the rate of false negatives and false positives.

Gupta and Gupta [26], developed CSSXC tool to enhance the detection of the XSS vulnerability in the web application that is deployed in the cloud environment. The tool uses the Sanitization mechanism to discover the hidden vulnerable point and detect the XSS attack by referring the blacklist of JavaScript vectors from the free available XSS repository.

Khan et al. [27], proposed an interceptor between the client side and server side to detect unknown malicious script code and prevent the XSS attacks by using machine learning classifiers such as support vector machine (SVM), k nearest neighbor (K-NN), J48 and Naïve Base and using the Wappen feature extractor.

The strengths and weakness of the existing XSS detection and prevention techniques are shown in table II while the comparison between these techniques in terms of the exploitation location, the attack types and the XSS taxonomy is shown in table III.

TABLE II: SUMMARY OF DETECTION AND PREVENTION TECHNIQUES OF XSS ATTACKS (2011-2017)

Authors	Deployment location	Tool	Strengths	Weakness	Dataset
Shahriar and Zulkernine, 2011 [18]	Web server	S^2XS^2	- Detect the XSS attacks automatically from server-side without any modification in the client entities or in the server. - The evaluation is performed in the real world with the zero false negative.	- This approach consumes large time in policy checks; thus the attack detection capability is low. - This approach includes false positive alerts.	JSPBlog, EasyJSP, MeshCMS and EmployDirectory
Shahriar and Zulkernine, 2011 [19]	Server-Side	prototype tool	- No need any modification in server script or browser - It does not execute JavaScript code to detect injected code.	This technique cannot discover each malicious injection code. It requires to remove the return keyword in event handlers and the comment tags.	JSP programs (JVote, Jauction and Jinsure)
Barhoom and Kohail, 2011 [20]	Server-Side	Detection solution	This solution can detect any injected JavaScript code breaks the input schema rules	- The huge number of requests is loaded from the server which reduces the performance of the open network.	http://ha.ckers.org/xss.html but this is fake website
Shar and Tan, 2012 [21]	server-side	SaferXSS	Detecting and preventing the client-side and server-side against the XSS attack in the real time	- Does not detect or prevent DOM-based XSS. - Analyses only the server-side - Uses only the OWASP Rule#1-Rule#5, thus any other untrusted context cannot be removed. - Targeted only Java-based web applications. - Required to modify the source code of the program instead of rewriting program bytecode, which is ineffective when the source code of the application is not available; thus the tool could not detect any vulnerable in this code. - Doesn't track information flow across the web page.	Events and Classifieds, download from http://www.gotocode.com Roomba, PersonalBlog, and JGossip download from http://www.sourceforge.net/ "
Gundy and Chen, 2012 [22]	Web browser and web server	Noncespaces	- Prevented the exploitation of XSS vulnerabilities. - Allow the client to differentiate between the trusted content created by web application and the untrusted content provided by the attacker	Doesn't contain any defensive mechanism for inserted JavaScript code when downloading from the remote web site.	NSmarty templates - TikiWiki case study - XSS Cheat Sheet
Huajie Xu et al., 2013[23]	Client side	Defensive model	- Provided effective protection for the client against XSS attack. - Support using different technologies e.g. PHP, ASP… etc. in the website. - Protect the user which access the website that have XSS attack	- It depends on the expectation behavior of the websites or the browser which limits its capability. - False negative alerts.	Source code from any website
Parameshwaran et. al., 2015 [24]	Client side	DEXTERJS	- Detects hundreds of Zero-day DOM XSS exploits in its benchmark. - zero false negative rates - Tested in the real world.	- Does not detect the original functionality of the client-side application that may include malicious code.	Alexa top 1000 sites
Guptaa and Gupta,2015 [1]	Server side	Automated detection system	- Tested on a PHP web application in the real world.	- A false negative rate needs improvement. - Does not detect the client side DOM-based XSS attack.	XSS Cheat Sheet, BlogIt and BloggIt
Gupta and Gupta, 2016 [25]	Cloud	Framework	- Does not require web browser modification. - Does not require the browser source code of the web application. - Low rate of false negative and false positive.	- Does not support the online social network (OSN).	OsCommerce PhpBB v2 XSS cheat sheet
Gupta and Gupta, 2016 [26]	Cloud	CSSXC	- High rate of true positive and low rate of false negatives. - Tested in the real world.	- This technique does not support the web application's OSN.	BlogIt, OsCommerce, Scarf, and Wackopicko XSS cheat sheet
Khan et al., 2017[27]	Client side	Interceptor	Detects unknown malicious JavaScript code from the client side.	- Usually 55 seconds are required to open a web page but this technique requires 97 seconds to open it.	leakiEst Malicious JavaScript

The previous techniques are solved some of the XSS attack problems to protect the user credentials on the web application, but the web application still has vulnerability contains a large amount of the sensitive data are lost

TABLE III: COMPARISON OF XSS DETECTING AND DEFENSIVE TECHNIQUES

Authors	Exploitation location	Year	Stored XSS	Reflected XSS	DOM-based XSS	Detection	Prevention	Server-side	Client-side
Shahriar and Zulkernine [18]	Web server	2011	✔	✔	✗	✔	✗	✔	✗
Shahriar and Zulkernine [19]	Server-Side	2011	✔	✔	✗	✔	✗	✔	✗
Barhoom and Kohail [20]	Server-Side	2011	✔	✗	✗	✔	✔	✔	✗
Shar and Tan [21]	server-side	2012	✔	✔	✗	✔	✔	✔	✗
Gundy and Chen [22]	Web browser and web server	2012	✔	✔	✗	✔	✔	✔	✗
Huajie Xu et al. [23]	Client side	2013	✗	✗	✔	✔	✔	✗	✔
Parameshwaran et al.[24]	Client side	2015	✗	✗	✔	✔	✗	✗	✔
Guptaa and Gupta[1]	Server side	2015	✔	✔	✗	✔	✗	✔	✗
Gupta and Gupta[25]	Cloud	2016	✗	✔	✗	✔	✗	✔	✗
Gupta and Gupta[26]	Cloud	2016	✗	✔	✔	✔	✔	✔	✔
Khan et al.[27]	Client side	2017	✗	✗	✔	✔	✔	✗	✔

The Stored-XSS attacks still need more research in order to enhance the detection and prevention of them. Also, the web pages are still vulnerable to the DOM-based XSS attacks which require from the researcher to develop new techniques for preventing this kind of threat.

IV. Conclusion and Future Works

The XSS attacks are still exploiting the web application vulnerabilities to steal the user credential. The techniques that are used to detect and prevent the XSS attack still needs more work to enhance the accuracy of XSS detection and prevention. Recently, OWASP developed a Web Application Firewall (WAF) model that can detect and prevent the Reflected XSS attack but the Stored XSS attack and the DOM-based XSS attack are still requires more work.

The future work is to develop a defensive mechanism that uses data mining and machine learning techniques, to detect and prevent the Stored XSS attack and DOM-based XSS attack in order to reduce the false negative and false positive.

References

[1] Shashank Gupta and B. B. Gupta, “Automated discovery of JavaScript code injection attacks in PHP web applications”, International Conference on Information Security & Privacy (ICISP), Nagpur, INDIA, 11-12 December 2015, Elsevier, Procedia Computer Science, vol. 78, pp.82 – 87, 2016.

[2] Gopal R. Chaudhari, Prof. Madhav V. Vaidya, “A survey on security and vulnerabilities of web application”, International Journal of Computer Science and Information Technologies (IJCSIT), ISSN: 0975-9646, Vol. 5 (2), pp. 1856-1860, 2014.

[3] Katy Anton, Jim Manico, and Jim Bird, “Top 10 proactive controls 2016”, OWASP, US, 2016.

[4] Jeff Wiliams and Dave Wichers, “Top 10-2017 rc1”, OWASP, US, June 30, 2017.

[5] Almudena Alcaide Raya, Jorge Blasco Alis, Eduardo Galán Herrero and Agustín Orfila Diaz-Pabón, “Cross-Site Scripting: An overview”, Innovations in SMEs and Conducting E-Business: Technologies, Trends, and Solutions, chapter 4, pp. 61-75, BN13: 9781609607654, 2011.

[6] Shashank Gupta.B.b Gupta, “Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art”, International Journal of System Assurance Engineering and Management, Vol. 8, Supplement 1, pp. 512–530, Springer, January 2017.

[7] Nikita Gupta and Analyst, “Cross-Site Scripting (XSS) research and intelligence report”, IBM, December 15, 2014.

[8] Acunetix, “Cross-site Scripting (XSS) attack”, HTTPS://WWW.ACUNETIX.COM/WEBSITESECURITY/CROSS-SITE-SCRIPTING/, [accessed: August 20,2017].

[9] Isatou Hydara and colleagues, “Current state of research on cross-site scripting (XSS) – A systematic literature review”, Information and Software Technology, No. 58, pp.170-186, Elsevier, 2015.

[10] WhiteHat security, “Applications security statistics report”, WhiteHat, Vol. 12, 2017.

[11] Cyber Security Review, “Bug in Magento puts millions of e-commerce merchants at risk of takeover”, http://www.cybersecurity-review.com/bug-in-magento-puts-millions-of-e-commerce-merchants-at-risk-of-takeover, [accessed: March 6, 2017].

[12] IT-SECURITYGUARD BLOG, “Sleeping stored Google XSS awakens”, https://blog.it-securityguard.com/bugbounty-sleeping-stored-google-xss-awakens-a-5000-bounty/ , [accessed: June 2, 2017].

[13] Security week, “XSS flaw exposed eBay users to phishing attacks”, http://www.securityweek.com/xss-flaw-exposed-ebay-users-phishing-attacks, [accessed: July 8, 2017].

[14] HTTPS://NAKEDSECURITY.SOPHOS.COM/2016/01/13/EBAY-XSS-BUG-LEFT-USERS-VULNERABLE-TO-ALMOST-UNDETECTABLE-PHISHING-ATTACKS/, [accessed: April 18, 2017].

[15] Monika Rohilla, Rakesh Kumar and Girdhar Gopal “XSS attacks: analysis, prevention & detection”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), ISSN: 2277 128X, Volume 6, Issue 6, June 2016.

[16] Ishikawa, Tomohisa, and Kouichi Sakurai. "Parameter manipulation attack prevention and detection by using web application deception proxy", Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, ACM, Jan 2017.

[17] Muhammad Parvez, Pavol Zavarsky and Nidal Khoury, “Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities”, 10th International Conference for Internet Technology and Secured Transactions (ICITST-2015), IEEE, 2015.

[18] Hossain Shahriar and Mohammad Zulkernine, “S^2XS^2: A server side approach to automatically detect XSS attacks”, 9th international conference on dependable, automatic secure computing. IEEE, pp. 7-17, December 2011.

[19] Hossain Shahriar and Mohammad Zulkernine, "Injecting comments to detect JavaScript code injection attacks.", Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, IEEE, pp. 104 -109, 2011.

[20] Tawfiq S. Barhoom and Sarah N. Kohail, “A new server-side solution for detecting Cross Site Scripting attack”, International Journal of Computer Information Systems, Vol. 3, No. 2, 2011.

[21] Lwin Khin Shar and Hee Beng Kuan Tan, “Automated removal of Cross Site Scripting vulnerabilities in web application”, Information and Software Technology, No. 54, PP. 467- 478, Elsevier, December 28, 2011.

[22] Matthew Van Gundy and Hao Chen, “Noncespaces: Using randomization to defeat cross-site scripting attacks”, Computers & Security, No. 31, pp. 612 – 628, Elsevier, 2012.

[23] Huajie Xu, Xiaoming Hu, and Dongdong Zhang, “A XSS defensive scheme based on behavior certification”, Applied Mechanics and Materials, Vols. 241-244, pp. 2365-2369, USA, 2013.

[24] Inian Parameshwaran, Enrico Budianto and Shweta Shinde, “DEXTERJS: robust testing platform for DOM-based XSS vulnerabilities”, 10th Joint Meeting on Foundations of Software Engineering(August 30-September 4), pp. 946-949, Bergamo, Italy, 2015.

[25] Shashank Gupta and B.B.Gupta, “Enhanced XSS defensive framework for web applications deployed in the virtual machines of cloud computing environment”, (ICETEST- 2015), Procedia Technology, No. 24, pp. 1595-1602, Elsevier, 2016.

[26] Shashank Gupta and B. B. Gupta, “CSSXC: Context-Sensitive Sanitization Framework for web applications against XSS vulnerabilities in cloud environments”, Procedia Computer Science, No. 85, pp. 198-205, Elsevier, 2016.

[27] Nayeem Khan, Johari Abdullah and Adnan Shahid Khan, "Defending malicious script attacks using machine learning classifiers", part of Ph.D., Malaysia Sarawak University, Hindawi, 2017.