Arraid Journal of Science and Technology
Higher Institute of Science and Technology - AShmokh

مجلة الرائد للعلوم والتقنية
المعهد العالي للعلوم والتقنية ـ الشموخ

Exploitation of Cross-Site Scripting Vulnerability on Web Applications

Mustafa A. Abo Mhara, Abdalla A. Abdarrahman[*], Suleiman O. Barnous[*], Adel S. Elashheb[**], Haytham F. Dhaw[***]

Faculty of Economics and Political Science, Bani Waleed University, Libya
[*]Higher Institute of Engineering Technologies Baniwalid, Libya
[**]College of Civil Aviation and Meteorology Espiaa, libya
[***]Higher Institute of Medical Technology Baniwalid, Libya
Contact E-mail: abdalla.1984.alosta@gmail.com

ABSTRACT
Attacks on web applications are increasing quickly with the advent of new technologies. Cross-Site Scripting (XSS) is a known vulnerability that exist in web applications. Attackers can exploit XSS to access web browser resources, such as cookies, credentials, and other sensitive information by injecting malicious client-side scripts into a website application. These scripts are then executed by users as they browse the site. This paper introduces XSS attacks with some examples of popular attacks. Methodologies for detecting and preventing cross-site scripting vulnerabilities are established and summarized by proposing a model for exploiting XSS vulnerabilities through reflected and stored attacks. The subsequent results are discussed and countermeasures are carried out to detect and prevent XSS vulnerabilities.

KEYWORDS: Web Applications, Cross-Site Scripting (XSS) Attack, JavaScript**.**

1. INTRODUCTION

Cross-Site Scripting (XSS) attack is a common vulnerability which is being exploited in web applications throughout the insertion of HTML tags and some sort of Java Script code. A poor input validation process on the web applications leads to steal cookies from the victim's web browser. Cookies are the mainly common technique to recognize and authenticate the users and it is being supported by almost all the web browsers [1].

XSS generally leads the most large increase web application vulnerabilities lists (e.g. OWASP, WhiteHat Website Security Statistics Report) [2]. XSS are in general classified into two main attacks that are Persistent and Non-Persistent Attacks [3].

Persistent attack holes are present when an attacker drive the malicious code on the vulnerable web application's storage area. As a result, if the stored malicious code gets used by the victim's browser, then stored attack gets exploited on the victim's web browser. Secondly non-persistent attack or reflected attack is that the vulnerable malicious code is not persistently stored on a web server however it is directly displayed by the vulnerable web application reverse to the victim's browser. Thus, the malicious code gets used on the victim's browser and finally the victim browser's assets have been compromised by the attacker.

Web applications are continuously has significant due to the augmented progress in the area of communication and information technology which caused to increase the number of users of Internet applications [4]. So, anyone is able to apply such applications or entrance confidential or unrestricted information at any time and from any place when the internet link is offered [5].

During this advancement and presented likelihoods of transmission of user's data of such applications, the problem of existing some vulnerabilities is come up such as SQL Injection and XSS attacks which permit hackers to exploit and take the important data or information through using web mechanisms such as cookies, sessions, or virus injection that are executed on the victim machine and becomes compromised by the hacker [6]. Therefore, the lack of securing Internet applications leads to limiting and minimizing their use.

The XSS is one of the critical holes in Web applications according to the top ten weakness of OWASP organization [7]. This kind of holes leads to serious penetration security similarly on the web site or a user where an attacker injects a malicious code to be executed later on by the user. Hence, if there is no auditing for the entered data, then the private user's date may be accessed by the attacker.

The structure of this paper is organized as follows: Section 2 introduces the background and related work on XSS attacks. Section 3 presents the proposed model of the XSS attack. Section 4 shows the experimental work and results. Section 5 describes the improvement and counter measures of XSS technique. Section 6 concludes the paper and introduces some recommendations for the future work.

2. BACKGROUND AND RELATED WORK

The XSS attacks come in various forms and can be divided into three categories. Reflected attacks, Stored attacks, the last type is DOM based attacks. We will survey each category of XSS in turn [8].

- Non-persistent attacks: this is the frequent variety of cross-site scripting exploit which is also call reflected attacks. It's a type of code injection [9]. This injection does not locate on the server. But arises when the victim loads a particular URL. The attacker uses social engineering and using methods of a coding link in order to deceive the victim to seem as trusted URL, to trick the victim to click on the URL to be displayed on the page contains a malicious script code [10]. If present, then the malicious code gets implemented on the victim's browser and finally the victim's browser resources have been compromised by the attacker.
- Persistent attacks: also known as (Stored XSS attacks) is an attack that has more risk than the reflect attacks because the code injection is stored at the server typically in a database and executed in the browsers of all users who use the web application [11]. This attacks usually post the malicious code in the comment field to be seem by other users later. In general attacks used to take control of the victim's browser, capture information on their applications, make website defacements, etc.
- DOM-Based XSS Attack: it is a kind of cross-site scripting attacks which is triggered at the client side machine, DOM based attacks using JavaScript to embedded malicious code and execute at the client-side later without sending information to the server-side to reference a session cookie or a form field [12]. This

attack based on the Document Object Model (DOM) of the page. A hacker imbedded the malicious code and modify the structure of HTML or XML. A hacker created crafted URL containing JavaScript. While the victims request crafted URL containing malicious code of JavaScript, When the user's browser processes server's response, the embedded JavaScript is executed.

3. THE PROPOSED MODEL FOR XSS EXPLOITATION

In this model, we paying attention on the exploitation of reflected and stored attacks. The attacker will check for an exploitation in the website. The first case if a reflected exploit is discovered the attacker will craft an URL and deliver this crafted URL to the victim to enabling the attacker to steal the victim information like stealing cookies. In the second case if a stored exploit is discovered the attacker will inject a malicious code typically into the database enabling the attacker to gain remote access to victim, as shown in figure 1.

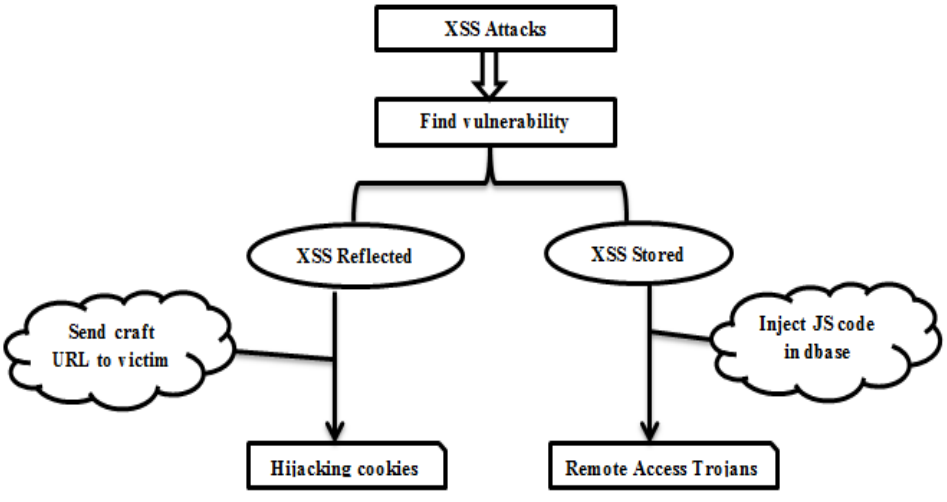

Figure 1: The proposes Model.

4. THE EXPERIMENTAL WORK

In this work, we used a local area network that consists of two work stations with Windows 10 operating system and a server connected by switch as shown in figure 2. One of work stations is used as a victim on machine and the other as a hacker as a server.

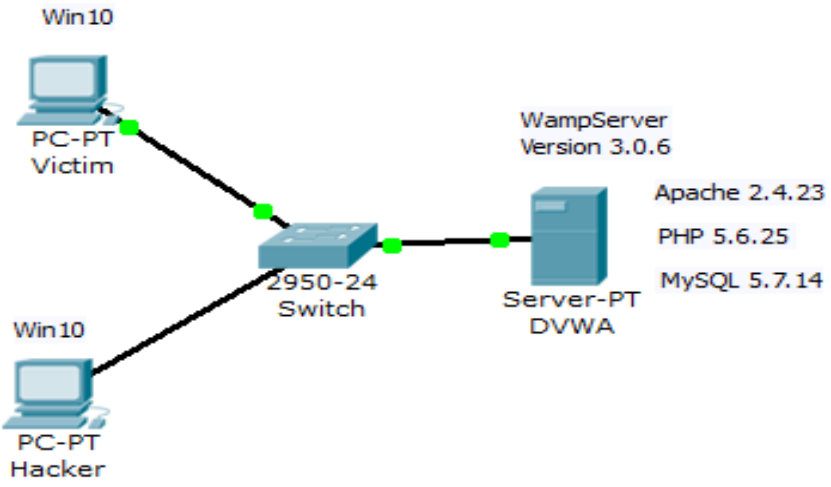

Figure 2: The used local area network.

4.1 Implementation of XSS Common attacks

In this work, we used the DVWA application to test reflected and stored XSS attacks, with change the security level at medium level as shown in figure 3, at this level any text has script tag will be remove.

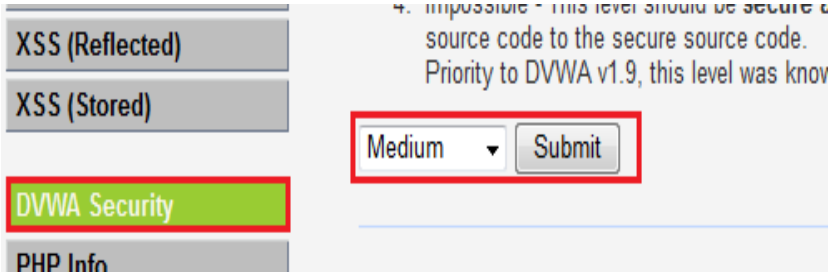

Figure 3: Testing reflected and stored XSS.

4.2 Reflected XSS attack

In this vulnerability, we injected malicious code using JavaScript to detect whether the site contains an XSS vulnerability. A simple JavaScript code to detect the vulnerability:

```
<script>alert("XSS ATTACKS")</script>
```

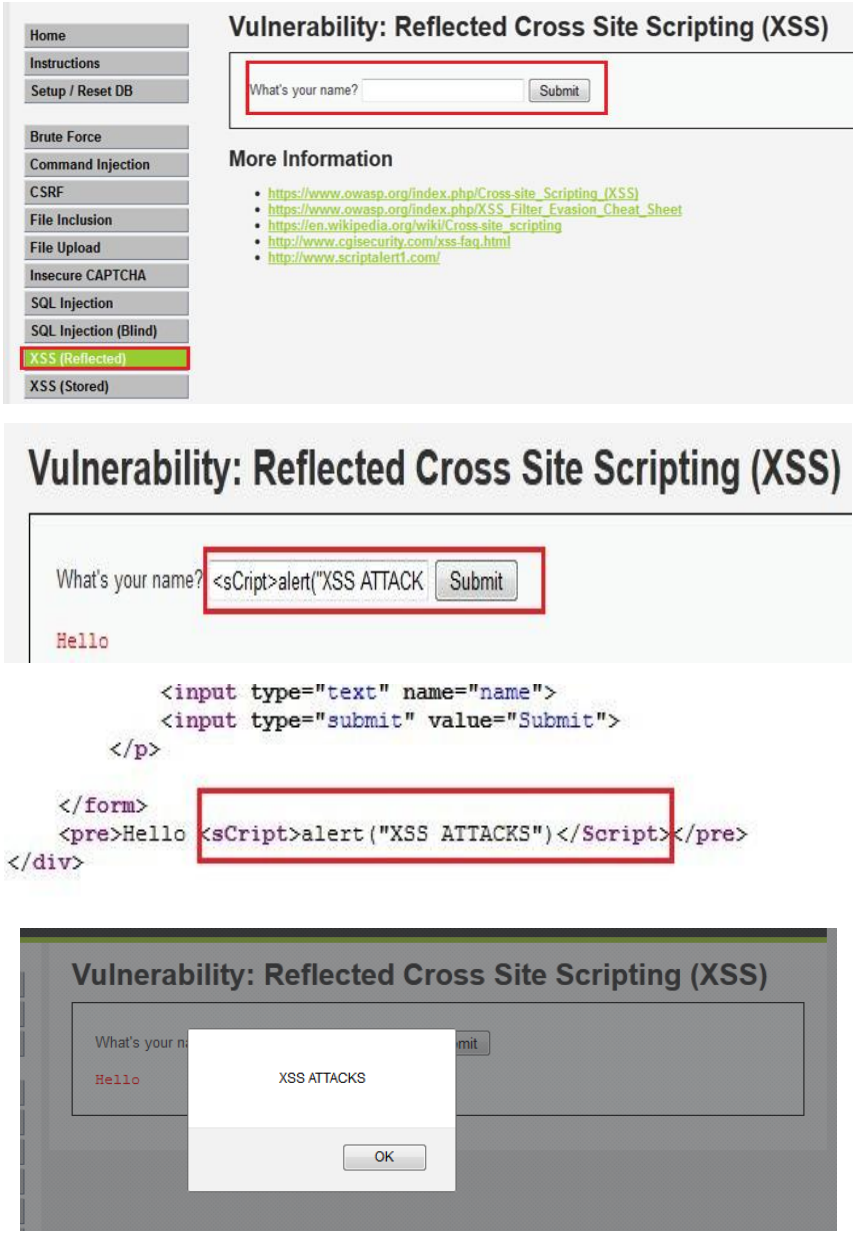

Figure 4: Detecting XSS Vulnerability.

The code to steal the victim's session and the testing result is shown in figures 4 and 5.

```
<imgsrc=x onError= alert(document.cookie)>
```

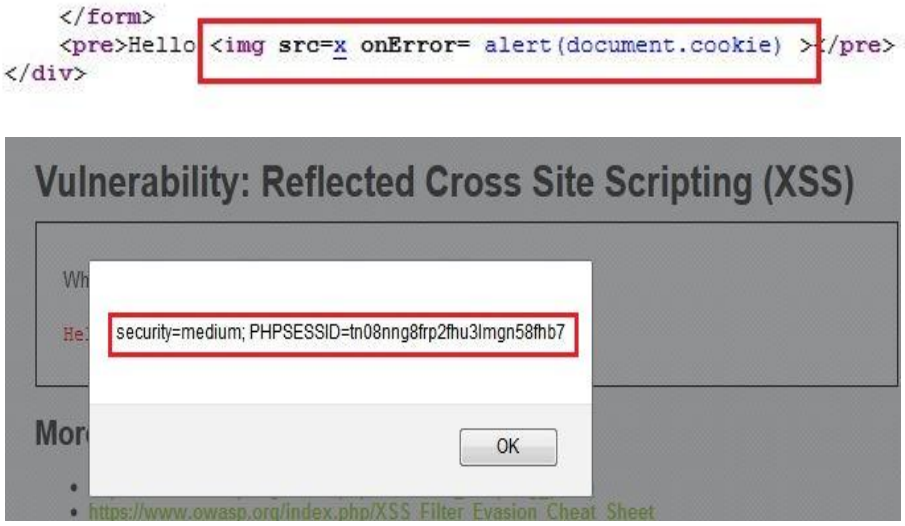

Figure 5: Detecting XSS Vulnerability.

The danger of this type of vulnerabilities does not only lie in stealing sessions, as they can be used to access important data such as a visa or bank account. Health can also be exploited and used in a denial of service (DOS) attack. This exploit is carried out by sending a suspicious email to the victim or redirecting him to a fake page containing a malicious link.

4.3 Stored XSS attack

We used software to send a virus to the health system in order to open a backdoor and control the victim's device remotely. The code below will send a file to the victim shown in figure 6.

```
<script>window.location='localhost/JRAT.exe;</script>
```

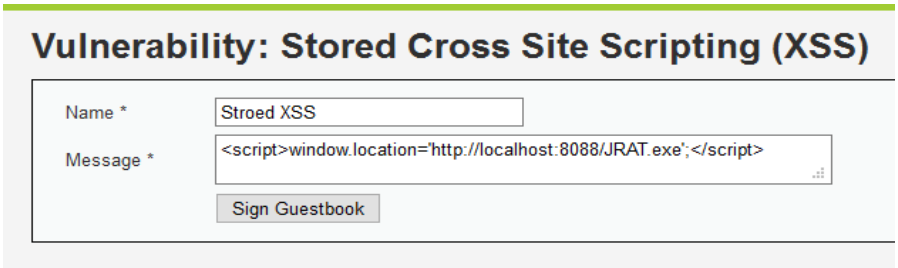

Figure 6: Stored XSS Vulnerability.

As shown in figure 7, once it is opened, an IP address will be sent and a port will be opened on the victim's device, enabling the attacker to take full control of the victim's device remotely.

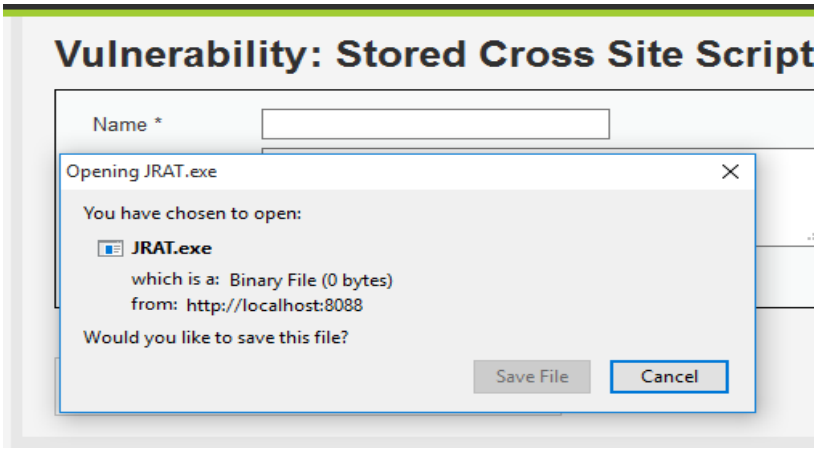

Figure 7: Stored XSS Attack.

This code will work automatically when any user enters the site because this type of vulnerabilities are stored on the server.

5. COUNTERMEASURES

We must take into account that the user will inject malicious code and attack Internet applications that allow the user to enter data. Therefore, we must prevent this attack using the proper strategies. For example, but not limited to, we can validate the input (input justification) that the user will enter. And cleanse the input of any malicious code. If the input type is numbers only, letters are not acceptable to overtake through. So, using output escaping and delete any JavaScript characters such as the word "script". Using the latest update of the internet browser and use some add-ons such as NoScript that limit XSS attacks. Use a specialized Web Software Firewall that will act as an further layer of guard beside these vulnerabilities. At the end, particular programs can be used to detect XSS vulnerabilities, such as the well-known project software (OWASP).

6. CONCLUSIONS AND FUTURE WORK

Cross-Site Scripting is a simple technique for attackers to access user's private information. In this research paper, we have summarized methodologies that previously defined for detecting and preventing cross-site scripting vulnerabilities. Moreover, by using the proposed model and its results, the XSS intrusion was exploited via HTTP and cross-platform properties to take and misuse the user's secret and important information. Hence, in order to avoid the vulnerabilities of XSS attack, we should take care about vulnerabilities of website applications. So, proper countermeasures should be done such as input validation procedure.

Furthermore, still cross-site scripting vulnerabilities remains a big trouble and challenge in web applications. That is due to attackers come across loopholes to pass up security methods should be followed by web developer. Therefore, there is no single solution that can successfully mitigates cross-site scripting attacks, and hence further research work is required and can be done to give entire solution against such sort of web vulnerabilities.

REFERENCES

[1] D. Kristol, “HTTP State Management Mechanism” in Internet Society, 2000. http:// www.ietf.org/rfc/rfc2965.txt

[2] Open Web Application Security Project: https://www.owasp.org/index.php/Top_10.

[3] S. Saha, “Consideration Points: Detecting Cross-Site Scripting,” (IJCSIS) International Journal of Computer Science and Information Security,Vol. 4, No. 1 & 2, 2009.

[4] Ande, R., Adebisi, B., Hammoudeh, M., & Saleem, J. Internet of Things: Evolution and technologies from a security perspective. Sustainable Cities and Society, 54, 101728, 2020.

[5] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660, 2013.

[6] Sarmah, U., Bhattacharyya, D., & Kalita, J. K. A survey of detection methods for XSS attacks. Journal of Network and Computer Applications, 118, 113-143, 2018.

[7] Paudel, S. Vulnerable Web Applications and how to Audit Them: Use of OWASP Zed Attack Proxy effectively to find the vulnerabilities of web applications, 2016.

[8] Rodríguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166, 106960, 2020.

[9] Gupta, S., & Gupta, B. B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. International Journal of System Assurance Engineering and Management, 8, 512-530, 2017.

[10] Heartfield, R., & Loukas, G. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. ACM Computing Surveys (CSUR), 48(3), 1-39, 2015.

[11] Mitropoulos, D., Louridas, P., Polychronakis, M., & Keromytis, A. D. Defending against web application attacks: approaches, challenges and implications. IEEE Transactions on Dependable and Secure Computing, 16(2), 188-203, 2017.

[12] Kirda, E., Jovanovic, N., Kruegel, C., & Vigna, G. Client-side cross-site scripting protection. computers & security, 28(7), 592-604, 2009.