Security Attack Vulnerabilities of WordPress

**Cross-Site Scripting Attack (XSS)**

XSS is a type of attack that is commonly experienced on the dynamic web pages. According to the attack, the attacker comes up with some malicious link, which is intended to execute unwanted script code on the website. Once the victim clicks the link, the code performs the malicious activities on the pages, such as sending the client’s cookie to a different server, modification of the affected site, stealing of the usernames and passwords, phishing, and form injection among many others (Nikiforakis 87).

Usually, vulnerability is common in web pages that use GET variables without proper filters or checks in place. For a long period of time, preventing the XSS attacks has been challenging for many developers because dynamism is inevitable in modern sites. However, the key prevention approach is embracing best data validation approaches. The validation should be practiced at both the client-side and the side for effectiveness (Bates, Barth and Jackson 91).

XSS vulnerability has been common to most of the WordPress versions, especially in versions 4.9.5 and lower. For example in WordPress 4.9.5, the vulnerability was in the get\_the\_generator function, where the version string was not escaped, thus, increasing the chances of attack through XSS. In version 4.9.2, the attack came about from the Flash Fallback files within the MediaElements located under wp-includes/js/mediaelement directory. In version 4.9.1, there were incorrect RSS and Atom Fields enclosure restrictions, increasing the chances of the attacks through the URL. Other lower versions had vulnerabilities in different PHP files, and the fixation was done with each release after realizations. Therefore, the versioning element of WordPress has played a critical role towards curbing the possible XSS loopholes. However, it is clear that it is impossible to make the judgment that the attack has been fully curbed because of its nature, where it is possible to happen in any function or PHP file (CVE Details).

**SQLI**

The SQL injection refers to a kind of attack that utilizes the SQL codes and queries for the backend database manipulation to get the ability to the access to the website of the database. When vulnerable, the attacker can utilize the CRUD (Create, Read, Update and Delete) operations in SQL to do engage in malicious activities to the site’s data. Usually, the injection capitalizes the cases where the application uses the untrusted data, especially the one entered in the form fields through the database queries. Applications that fail to make a proper data sanitation increases the chances of the SQLI attacks because they will take in the SQL commands which will later affect its data (Martin and Lam 36).

Researchers have found that the best approaches towards controlling the SQLI attacks and other online vulnerabilities are through the utilization of the Web Application Firewall (WAF). Usually, WAF is implemented basing on the list of the possible signatures that detects the attacks. The list is regularly updated through the collection of the possible SQLI vulnerabilities and fixing them. It introduces the blocking rules that are embraced to overcome these challenges for effectiveness (Halfond, Orso and Manolios 65).

WordPress, just like other frameworks and systems, has also faced been vulnerable to some SQLI attacks in some of its versions. Usually, once the WordPress attack has been identified it is fixed in the later version release. Some of the SQLI vulnerabilities were detected in WordPress Version 4.8.3, where the $wpdb->prepare() function was vulnerable to the attack. Also, version 4.8.2 had a similar vulnerability but on the wp\_signups.activation\_key function. In the same version, a mismatch in the % characters had exposed the $wpdb->prepare function to be vulnerable to the attack especially through the themes and the plugins. Other lower versions had also experienced similar vulnerabilities but were fixed depending on the vulnerable file (CVE Details).

**The Cross-Site Request Forgery (CSRF)**

The CSRF attack is one of the common vulnerabilities to modern dynamic websites. Usually, the attack capitalizes on the user’s trust to the site, by tricking them to reveal sensitive data. The attack makes the users perform activities that they do not intend to do in the legitimate web systems. Once tricked, the users are forced to authenticate themselves and thus, reveal their usernames and password, which are later used for malicious purposes. They identify the rights and privileges possessed by a given user and use them to perform malicious activities. Other similar attacks are based on the cookies where the attacks use this information to do bad activities (Barth, Jackson and Mitchell 77).

Researchers have established that the best way to control the CSFR attacks is through the prevention of the attacks on the server and on the client side. On the server side, it is critical to ensure that there is a clear approach towards managing the transitions of cookies. Session-tracking tokens and activities are important and should be managed effectively through the dynamic generation. The approach is effective because it makes it difficult for the attacker to perform malicious activities. Also, researchers have established that the attack can be reduced by enlightening the users about the attacks. They need to reduce their trust in the sites that appear dubious and vulnerable. They must be careful on the pages or the activities where they submit their login credentials for effectiveness.

The CSRF in WordPress was last noted and fixed in version 4.7.5 where it existed in the file system credential dialogue, and it did not even require a nonce to update the credentials. Also, the attack vulnerability had been noted in version 4.7.3, on the wp-admin/includes/class-wp-press-this.php file, where it could cause excessive usage of the server resources. Through the CSRF attacks, there was a high possibility of triggering the outbound HTTP request for the large files parsed. Lower versions had also been prone to the attack especially through the use of plugins and themes (CVE Details).

**User Enumeration**

The user enumeration is a kind of attack where the attacker uses the techniques such as brute-force to make the guesses or the confirmation of the correct user credentials in the system. Usually, the attack happens at the forget password form or the login form where the attackers try to find a hint to enter the system. They also capitalize on server responses to derive a confirmation on the validity of the credentials that they guess (Qi 39).

The scientists have suggested various ways that can help in prevention of the user enumeration attacks. They include the implementation of the graceful sign-out errors, which do not give the attackers a hint about the guessed credentials. Also, the use of log out mechanisms for the account is important, and helps to reduce the attacks. There should be a limited number of login attempts, and lock the account for some period of time if the activities are persistent. Besides, the IP blocking mechanisms can be used to reduce the attack. Additionally, it is important to consider the forgot-password activities to solely rely on the user information and not the stored cookies.

In WordPress, the vulnerability of the user enumeration was last experienced in versions 4.7 and 4.7.1 in the wp-includes/rest-api/endpoints/class-wp-rest-users-controller.php file and the REST API. The framework could not properly restrict the list of authors of the post. The loophole gave the attackers the possibility to access sensitive information on the site through making the requests via the wp-json/wp/v2/users. However, the issue was resolved in later versions (CVE Details).

**Privilege Escalation**

The privilege escalation is a type of system attack where one uses the approaches of network intrusion to take advantage of the possible programming errors within the system or the possible design flow issues, to gain an elevated access to data, application or other resources within the system. Once they get the privileges, they use them maliciously in the system (Monshizadeh, Naldurg and Venkatakrishnan 670).

The best approach to overcome the privilege escalation attacks is through the use of the Privilege Access Management (PAM) strategies. The model provides an improved solution to possible attacks to the system, by providing control to both deliberate and accidental privilege misuse, thus improving the security. An effective PAM should contain access managers, password vaults and session managers, which work together for effectiveness. The role of the PAM in the security management includes granting of the privileges to the users that have been authorized. It also revokes access as needed for effectiveness. The model should also cover the aspects of controlling direct password access to the privileged users of the system. Besides, the model needs to have a centrally managed access to effectiveness and an audit trail to provide a mechanism for monitoring the operations of the privileged users (Djeric and Goel 355).

Earlier versions of WordPress were vulnerable to the privilege escalation attacks. Before version 4.8.2, WordPress was vulnerable to the traversal attacks in the directory, especially in the customizer. Also, there were aspects of the directory access through the unzip operations at the ZipArchive components at the earlier versions. Besides, there were loopholes in the bypass activities for the versions of 4.7.2 and below, where attackers could gain access because it could not protect the taxonomy effectively. Version 4.7.1 had access issues at Multiple WordPress API, where it could not generate random keys properly, thus, could not effectively implement the restrictions (CVE Details).

Works Cited

CVE Details. “Wordpress: List of Security Vulnerabilities”. *Cvedetails.Com*, 2018, https://www.cvedetails.com/vulnerability-list/vendor\_id-2337/product\_id-4096/. Accessed 3 June 2018.

Barth, Adam, Jackson, Collin and Mitchell, John C. “Robust Defences for Cross-Site Request Forgery.” *Proceedings of the 15th ACM Conference on Computer and Communications Security*, 2008, pp. 75-88.

Bates, Daniel, Barth, Adam and Jackson, Collin. “Regular Expressions Considered Harmful in Client-Side XSS Filters.” *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 91-100.

Djeric, Vladan and Goel, Ashvin. “Securing Script-Based Extensibility in Web Browsers.” *USENIX Security Symposium,* 2010, pp. 355-357.

Halfond, William, Orso, Alex and Manolios, Pete. “WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation.” *IEEE Transactions on Software Engineering*, vol. 34, no. 1, 2008, pp. 65-81.

Martin, Michael, and Lam, Monica S. “Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking.” *Proceedings of the 17th Conference on Security Symposium*, 2008, pp. 31-43.

Monshizadeh, Maliheh, Naldurg, Prasad and Venkatakrishnan, V. N. “Mace: Detecting Privilege Escalation Vulnerabilities in Web Applications.” *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 690-701.

Nikiforakis, Nick. “SessionShield: Lightweight Protection against Session Hijacking.” *International Symposium on Engineering Secure Software and Systems*, 2011, pp. 87-100.

Qi, Lianyong. “Combining Local Optimization and Enumeration for QoS-Aware Web Service Composition.” *IEEE International Conference*, 2010, pp. 34-41.