Summary on "A proposed approach for preventing cross-site scripting" by Taha and Karabata (2018)

Introduction

Website vulnerability can be defined as any form of flaws present in a website or a code developed for web application through which an attacker can influence the site. There are a range of security vulnerabilities present in websites with Cross site Scripting (XSS), SQL injection, security misconfiguration or broken authentication being the most common ones. These vulnerabilities can be exploited by hackers for purposes of stealing data, to deploy malwares or to hold the data for ransom. Attempts are continuously being made to prevent these exploits and increase web security. In this context, a paper which proposes an approach to prevent cross site scripting, published in 2018 by Taha and Karabata, has been selected for the present assignment. Firstly, a brief background to the techniques and technologies that are used for implementation of Cross Site Scripting will be provided in this assignment. Then a summary of the paper selected for the assignment will be presented followed by a list of references utilized.

Background

Cross site scripting, abbreviated as XSS, is a form of injection wherein the trusted and benign websites are injected with harmful codes. It is an attack on the web applications and websites where the privacy of the users of any specific site or browser can be breached completely leading to security issues as the details of the customers are controlled or stolen. The XSS attacks can be implemented through a range of client side languages including JavaScript, HTML, ActiveX, VBScript and Flash. The main technique used to implement XSS is to manipulate a vulnerable site's vulnerable script through a malicious code. The script of the site to be attacked generally includes a portion of the parameters for the HTTP request and gives the parameters back in the response page. The data to such script will contain information such as the access to cookies and the attacker will then manipulate it in a harmful manner by breaching the privacy of the client completely. This form of web exploit includes three different aspects: the user, the attacker and the website. The attackers' objective is to get any form of sensitive information from the user of the website and utilize that data in an illegitimate manner. Though the vulnerability to attack originates from the website, the direct harm is not caused to the website but to the user of that website.

Summary of the paper

The term "Web" was coined in 1989 by Tim Berners which refers to the World Wide Web, abbreviated as www, is a service which provides its users with access to web pages through browsers so that they can involve in data sharing. In the traditional HTML (Hyper Text Markup Language) format the websites were static and limited data exchange was allowed. But with commercialization of www and uses of other side languages the restrictions on interactions and data exchange was eliminated. However, the issue of attacks on websites has now emerged as a big threat in web security. There is a list of top web security risks and recommendations developed by Open Web Application Security Project (OWASP) which can be used by security

technicians as a measure to analyze the applications being used across web. Cross site scripting (XSS) is one of these threats in which an attacker can utilize a code injection attack in any of the client side script language, mostly in JavaScript, to carry out malicious web scripts in the browsers. The attack can be executed through any emails links or also through the attacker's web page. In 2017, XSS was ranked as the third most prevalent and dangerous web exploits in the web applications of healthcare institute with SQL injection and DoS attack being ranked as first and second respectively. Broadly, XSS can be categorized into two types: stored and reflected. Stored (or persistent) XSS is characterized by the permanent storage of harmful code on the servers of the targets. This is done through the identification of weak points in the target web application followed by the injection of the harmful code to cause damages. In reflected (non persistent) XSS, the attached code is not located on the web server but sent through links to the victims in the form of emails or redirected links on some other web servers. There is another type of XSS which is less common i.e. DOM XSS which relies on manipulation of a number of DOM objects on the web page so that the inappropriate processing creates the condition of XSS.

An example of XSS is a harmful code, developed by having HTML code of the browser parser of victim, which is created with an intention of stealing cookies information from the user of the victim. A code which guides the browser of the user to a different URL can be developed by the hacker in JavaScript and then access to the cookie information from the victim can be gained. Subsequently, the cookie information can be used to execute other attacks on the victim illegally. This type of XSS attack is a major threat happening in web applications as well as web sites. A number of researches have been dedicated in the past to mitigate, detect and prevent these security issues. Many solutions have been devised over time to prevent XSS such as the development of tools which characterizes the arrangement for development of a safe website, strategies to make all stages of application life cycle secure, use of Microsoft ASP. Net platform and content filtering. One of the priorities to prevent XSS is the application of a context dependent output encoding so that the data provided by users can be validated and consistence can be maintained. Four different mechanisms can be suggested for validation of user inputs, namely replacement, removal, escaping and restriction. The replacement way identifies malicious inputs from users and replaces those with correct ones. The malicious inputs can be removed if the second mechanisms is to be followed. In the escaping mechanism, the main data characters can also be changed in a way that they cannot be interpreted by a malicious code for the escaping mechanism. In the restriction mechanism, the inputs from the users is checked to restricted non malicious ones. As per OWASP's guideline, it is also important to reject data identified as harmful, accept only the data that is known to be valid and clean any malicious data when dealing with user data.

The system proposed for detection and prevention of XSS utilizes two different methods based on secure code PHP functions. The first method validates data obtained from the user's web forms through utilization of regular expressions obtained while the second method is based on removal mechanism. In the second technique of prevention, an alternative regular expression is utilized for checking each of the inputs that may face a harmful script in it. If there is any injection of XSS script code, these harmful codes will be removed instantly. The efficiency of the proposed system was assessed through vulnerable PHP websites. The built in functions of the

PHP programming language can help in the prevention of XSS attacks occurring through htmlspecialchar() and htmlEntities (). By utilizing regular expressions found easily, these functions adjust characters to HTML entities while also substituting and working with string. The main strategy in this approach are two regular expression, namely AllowList, which works to validate only those trusted and anticipated user inputs, and DenyList, which examines if any unfitting data is present and removes if such suspicious characters are present. This system may also be a basis for working towards the development of extensions that can lead to automatic detection of harmful codes in the future.

To conclude, the paper discusses overall aspects of web security with an emphasis on XSS threats and its types. It discusses how regular expressions of a programming language can be utilized to protect a web page in an instance where any hacker attempts to run harmful code on the browser of the victims.

References

1. Cross-site Scripting (XSS) - OWASP. (2019). Retrieved 21 December 2019, from https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
2. Watchfire Corporation. (2006). Retrieved 21 December 2019, from http://index-of.es/Security/css-explained.pdf
3. Whitepaper, Sophos (2018). Retrieved 21 December 2019, from https://www.sophos.com/en-us/medialibrary/Gated-Assets/white-papers/Sophos-Comprehensive-Exploit-Prevention-wpna.pdf
4. Taha, T., & Karabatak, M. (2018). A proposed approach for preventing cross-site scripting. Retrieved 21 December 2019, from https://ieeexplore.ieee.org/document/8355356