Chapter 39 BDS: Browser Dependent XSS Sanitizer

Shashank Gupta
National Institute of Technology Kurukshtra, India

B. B. Gupta
National Institute of Technology Kurukshtra, India

ABSTRACT

Cross-Site Scripting (XSS) attack is a vulnerability on the client-side browser that is caused by the improper sanitization of the user input embedded in the Web pages. Researchers in the past had proposed various types of defensive strategies, vulnerability scanners, etc., but still XSS flaws remains in the Web applications due to inadequate understanding and implementation of various defensive tools and strategies. Therefore, in this chapter, the authors propose a security model called Browser Dependent XSS Sanitizer (BDS) on the client-side Web browser for eliminating the effect of XSS vulnerability. Various earlier client-side solutions degrade the performance on the Web browser side. But in this chapter, the authors use a three-step approach to bypass the XSS attack without degrading much of the user's Web browsing experience. While auditing the experiments, this approach is capable of preventing the XSS attacks on various modern Web browsers.

INTRODUCTION

In the recent times of World Wide Web (WWW), static web pages were used on the web server side for responding the request of various users. As the name signifies the sense, these web pages generally provides the content in response web page as it is stored on a web server. These web pages are suitable for such type of content that rarely or never needs to be updated. Though, static web pages had no major security concerns. However preserving large amount of static web pages on the web server side is unfeasible without automated tools. Therefore the performance of the web server with static web pages becomes a bottleneck.

On the other hand, in order to increase the readability and enhancement of the web page, the web application comes into the era of dynamic web pages. These web pages are designed in such a way to

DOI: 10.4018/978-1-5225-3422-8.ch039

meet the personalized and up to date information to the client. Dynamic web pages generally provide the web content in a response web page based on the user supplied input. Client's web browser sends the HTTP request by initiating an interaction with the web server. The web server will collect the data from various resources and assembled into one web page and finally transmit it to the web browser. There are many tools (like bulletin boards, search engines, login forms etc) available in the dynamic web pages which demands the interaction between the web server and user.

Apart from these useful features of dynamic web pages, these web pages also allow the attackers to embed the malicious code as well. When such types of malicious codes are executed by web browser, then the user has to compromise various types of credential resources (e.g. cookie, credit card numbers etc.). In the present era, the modern dynamic web application assures the security of some of the important credentials of a user (like user-id and password). However, there are some other sensitive credentials (e.g. cookies) which are the best possible targets for the attackers. If such credentials are compromised by an attacker, then he can simply hijack the victim's session, access the confidential resources of its web application etc.

Cross-Site Scripting (XSS) (Alfaro et al., 2007) is one of such vulnerability which is generally caused by the improper sanitization of user input. The web server processes the information without checking for the vulnerable code injected in it and displays this vulnerable content in the web page and sends it the web browser. The web browser executes this vulnerable code and transfers its credential resources to the attacker's domain. The main goal of XSS attack is to steal the cookies of the victim's web browser and redirect it to the attacker's web application. Later on the attacker can use this cookie to access the sensitive resources of the web application of victim.

Hyper Text Transfer Protocol (HTTP) (Gourley et al., 2002) is a client/server based internet protocol which is generally used for information exchange between the client and server. Although, it is a state-less protocol, therefore it does not store any information regarding the session initiated by the client and web server. It also does not guarantee that the HTTP requests and corresponding HTTP responses are shared by the same user or not. Therefore this protocol cannot prove the authenticity of the user using the internet. To overcome this problem, cookies comes into picture, which are simply text files generated on the web server side.

Cookie is generally shared between the web browser and web server. The main goal of sharing the cookie file between the web server and web browser is to maintain the continuity between them. The web applications generally use cookies to provide a mechanism for creating state full HTTP sessions. The cookies are supported by nearly all up to date web browsers and therefore allow for a greater flexibility in how user sessions are maintained by the web applications. For web applications that need authentication, they frequently use cookies to store the session Ids (Kristol, 2000) and then pass the cookies to the users after they have been authenticated. The cookies are stored in the user's web browser. The web browser returns the cookies every time it needs to reconnect as a part of an active session and then the web application associates the cookies with the user. The following Figure 1 shows the scenario of cookie sharing between the web browser and web server.

The following are the steps of interaction between client's web browser and server for exchanging HTTP request and response messages:

1. In most of the cases whenever a client wants to access the resources of web application, the first step is to submit a user-id and password through web browser to the web server of a particular web application.