ISSN 2249-6343
International Journal of Computer Technology and Electronics Engineering (IJCTEE)
Volume 2, Issue 3, June 2012

A Pattern Based Approach to Secure Web Applications from XSS Attacks

Dr R.P Mahapatra, Ruchika Saini, Neha Saini

Abstract— **As web applications must be available 24/7 and offer data access to customers, employees, suppliers, and others, they are frequently the weak link in enterprise security. So web applications are popular attack targets due to the lack of coordination and lack of security awareness on part of the developers. When hackers gain access to web applications, they often have direct access to confidential back-end data on customers and the company. Cross Site Scripting Attack belongs to top ten web application vulnerabilities. This paper proposes a mechanism to secure java web applications from XSS (Cross Site Scripting Attack) by applying a framework based on pattern matching approach. The proposed framework consist Request/Response Analyser and Modifier modules. The first interaction of request is to Request Analyser/Modifier Module which decides about request is malicious or not and takes decision accordingly. Response analyser and Modifier module deals with the data to be returned the client, it modifies the malicious response to harmless data. Attack Recorder and Response Rejecter Module records the malicious Request/Response for future use. Java Regex has been used for pattern generation and matching the malicious attack signatures. Some comparisons have been made between existing solutions and the proposed framework. We tested the pattern matching based framework on some java web applications. The strength of the framework is that it can be applied on any existing java web application without source code modification.**

Index Terms-**Pattern, Regex, Request, Response, XSS**

I. INTRODUCTION

Pages of dynamic Web Applications often contain user-supplied parts, when insufficiently filtered, malicious scripts can be injected along with these parts, assuming, that these are executed in a user's browser, it is possible to exploit the trust relationship between user and web server for instance, compromising authentication credentials. This type of attack is called Cross Site Scripting (XSS) Attack.

A. Types of XSS Attacks: There are three types of known XSS flaws [1],[2] and 3] Stored XSS, Reflected XSS, and DOM based XSS, among which Stored XSS vulnerability always allows the most powerful attacks. During this type of attack, attack vectors are submitted to web server and stored on the server (in a database, file system or other locations). When other users request this data and attack vectors are displayed on their browser without sufficient checks, an attack may happen. Hackers use XSS vulnerability to gain access to victims' browser for identification e-mail or password. Because the malicious script running under the context of current user, firewall, encryption method or IDS (intrusion detection system) are ineffective in preventing this type of attack. As described in OWASP TOP10 security risks 2010 [4], XSS is one of the most critical risks. Attackers often perform XSS exploitation by crafting malicious URLs and tricking users to clicking them. These links cause client side scripting languages (VBScript, JavaScript, etc.) of the attacker's choice to execute on the victim's browser. There are numerous ways to inject JavaScript (any script) code into URLs for the purpose of a XSS attack [5], [6], [7], [8] and [9].

B. Injection Points: Where can our web applications fall victim these are called injection points [10]. Since XSS works as an interaction with active server content, any form of input should be filtered if it is ever to show up in an html page. The default example, and the easiest to exploit, is parameters passed in through query string arguments that get written directly to page. These are enticingly easy because all of the information can be provided directly in a clickable link and does not require any other html to perform. Sites that are particularly vulnerable to this form of attack would include guest books, html chat rooms, message boards, discussion forms etc. Some servers include special "404 Page Not Found" or servlet error messages that detail the page that was requested, or parameters passed in. If these elements are not filtered they provide a perfectly overlooked breeding ground.