Volume 6, Issue 11, November 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Cross Site Scripting_Vulnerabilities and Defence

Monali Sachin Kawalkar
Department of Computer Science, R.T.M. Nagpur University, Nagpur,
Maharashtra, India

Abstract— Cross Site Scripting is a most prevalent web application security issue. This occurs when application sends the user provided data to the web browser without validating or encoding the account. XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites. In this technique hackers execute embedded malicious script on the client machine. The script executed could have the capabilities of reading, modifying or transmitting sensitive data Thus is lets attacker to execute script in the victims browser to hijack user sessions, defence web sites, insert hostile content, conduct phishing attacks, and take over user's browser. Mainly the sites that reflect back user inputs without validating the contents are prone to such attacks. The code containing malicious script is usually written in HTML/Java Script, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. All web application frameworks are vulnerable to this kind of attack.

Keywords— XSS – Cross site scripting, OWASP – Open Web Application Security Project (OWASP) , SQL – Structure Query language, WAF – Web application Firewall, WASC – Web application Source Code, DOM - Document Object Model

I. INTRODUCTION

According to the Open Web Application Security Project (OWASP) , cross-site scripting (XSS) is already one of the top two vulnerabilities. Cross site scripting attack is a class of web application vulnerabilities in which an attacker cause a victim's browser to execute JavaScript crafted by the attacker to gain elevated access privileges to sensitive page-content, session cookies, and a variety of other information.

These attacks can steal confidentiality of sensitive data, undermine authorization schemes, defraud users and defame web sites.

There are basically three large categories of XSS attacks, a) reflected , b) stored and c)DOM based.During a reflected XSS attack the injected code is placed in a URL, upon the user clicks on the malicious URL, the injected code executes. On the other hand, during a stored XSS attack, the adversary injects the malicious payload in some form of storages utilized by a web application.

II. WHAT IS CROSS-SITE SCRIPTING?

A cross-site scripting attack is a kind of attack on web applications in which attackers try to inject malicious scripts to perform malicious actions on trusted websites. In cross-site scripting, malicious code executes on the browser side and affects users. Cross-site scripting is also known as an XSS attack.

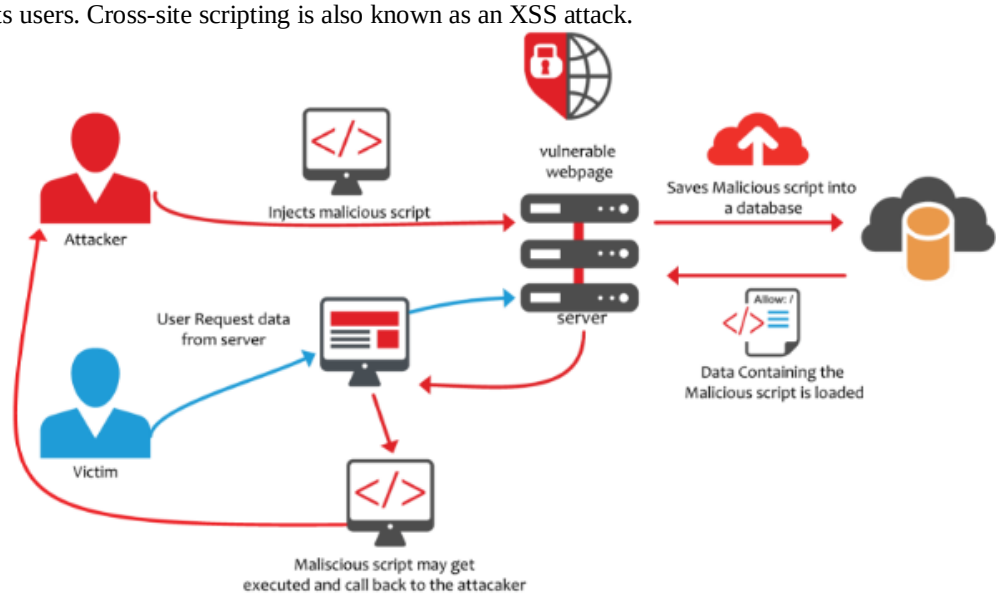

Figure 1

A cross-site scripting attack occurs when a web application executes a script that the attacker supplied to end users. This flaw can be found anywhere in an application where user input has been taken but not properly encoded. If the input is not properly encoded and sanitized, this injected malicious script will be sent to users. And a browser has no way to know that it should not trust a script. When the browser executes the script, a malicious action is performed on the client side. Most of the times, XSS is used to steal cookies and steal session tokens of a valid user to perform session hijacking. The above description is represented in below diagram (Figure 1)

III. EXAMPLE ATTACK SCENARIO

The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";

The attacker modifies the 'CC' parameter in their browser to:

'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi? foo='+document.cookie</script>'.

This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

Types of Cross Site Scripting Attack–

1. Non-persistent XSS

Non-persistent XSS is also known as reflected cross-site vulnerability. It is the most common type of XSS. In this, data injected by attacker is reflected in the response.

The non-persistent (reflected) XSS vulnerability results from the data commonly in HTTP query parameters or HTML form submissions which are provided by web client. The attacker uses social engineering to convince a victim to click on a disguised URL that contains malicious HTML/JavaScript code.

The user's browser then displays HTML and executes JavaScript which was a part of the attacker crafted malicious URL. This can result in stealing of browser cookies and other sensitive user data. To prevent firstorder.

XSS attacks applications need to reject or filter input values that may contain script code. Script is not stored but reflected as such to the users.

2. Persistent XSS

Script is stored and reflected back to same or different user.

Persistent cross-site scripting is also known as stored cross-site scripting. It occurs when XSS vectors are stored in the website database and executed when a page is opened by the user. Every time the user opens the browser, the script executes. Persistent XSS is more harmful that non-persistent XSS, because the script will automatically execute whenever the user opens the page to see the content

The persistent (stored) XSS vulnerability results from the application that store part of the attacker's input in a database, and then inserting it in an HTML page that is displayed to multiple victim users. It is harder to prevent stored XSS than reflected XSS for two reasons,

a) social engineering is not required, the attacker can directly supply the malicious input with tricking users into clicking on a URL,
b) a single malicious script once planted into a database can execute on the browsers of many victim users later. And applications need to use variety techniques to reject or sanitize input values that may contain script code which are used in database commands. Due to the prevalence of XSS attacks and current trend in web applications, there exists a strong need for preventing these attacks.

3. DOM Based

This type of attack targets the codes of the webpage itself. These attacks happen due to improper use of Document Object Model (DOM) by JavaScript. Older versions of Internet Explorer even allowed these attacks in the local pages.

As the codes can be altered before they are interpreted by the browser, it may seem that the malicious codes were sent from the server.

Impact

- Page redirection-This has the ability to redirect the readers to a different URL result in bad user experience and exploitation of trust relationship between the Application and the end user.
- Credential theft
- Denial of service
- Cookie theft and as a result User Impersonation

Counter Measure

- Disable scripting in the web browser and from e-mail clients
- Do not echo user input without proper sanitation.
- Encode all non-alpha numeric character in user input
- Enforce response length. Truncate if it goes beyond the specific limits.
- Educate users on the usage like not to visit sites through link provided in discussion forums or e-mails. Always open a fresh browser and type the URL in the address bar to visit the site.

● Use proper filter on user supplied data.

Let us start with basic filters:

There is a simple rule that you need to follow everywhere: Encode every datum that is given by a user. If data is not given by a user but supplied via the GET parameter, encode these data too. Even a POST form can contain XSS vectors. So, every time you are going to use a variable value on the website, try cleaning for XSS.

These are the main data that must be properly sanitized before being used on your website.

- The URL
- HTTP referrer objects
- GET parameters from a form
- POST parameters from a form
- Window. location
- Document. referrer
- document.location
- document.URL
- document.URLUnencoded
- cookie data
- headers data
- database data, if not properly validated on user input

First of all, encode all <, >, ‘ and “. This should be the first step of your XSS filter. See encoding below:

- & –>&
- < –><
  > –>>
- ” –>"
- ‘ –>'
- / –>/

A. “What can I do to protect myself as a vendor” –

Never trust user input and always filter metacharacters. This will eliminate the majority of XSS attacks. Cobverting<and> to <and> is also suggested when it comes to script output. Remember XSS holes can be damaging and costly to your business is abused. Often attackers will disclose these holes to the public, which can erode customer. And public confidence in the security and privacy of your organization site. Filtering <and> alone will not solve all cross site scripting attacks. It is suggested you also attempt to filter out (and) by translating them to, (and) ,”to&quote;, ‘to', and also # and & by translating them to&#35 and & (&).

B. “What can I do to protect myself as a user” –

The easiest was to protect yourself as a user is to only follow links from the main website you wish to view. If you visit one website and it links to CNN for example, instead of clicking on it CNN’s main site and use its search engine to find the content. This will probably eliminate ninety percent of the problem. Sometimes XSS can be executed automatically when you open an email, email attachment, read a guestbook or bulletin board post. If you plan on opening an email, or reading a post on a public board from a person you don’t Know BE CAREFUL. One of the best ways to protect yourself is to turn off javascript in your browser settings. In IE turn your security settings to high. This can prevent cookies theft, and in general is a safer thing to do.

IV. CONCLUSION

Cross-site scripting is one of the most dangerous website vulnerabilities. It is used in various ways to harm website users. Mostly it is used to perform session hijacking attacks.

We also know that patching XSS is possible but we can never be 100% sure that no one can break our filter. Hackers always find ways to break filter security. As a website owner or web developer, it is your responsibility to create a secure application that protects users’ data, so you must find and patch dangerous web application vulnerabilities.

Nevertheless, many service providers are either not willing or not able to provide sufficient protection to their users. To ensure protection against more subtle types of XSS attacks that try to leak information through Non Persistent XSS and persistent XSS use proper filters on user supplied data.

REFERENCES

[1] SQL Injection is not a dark art. A wealth of information is available on the internet regarding how to use it and how to protect against it.
[2] Www.OWASP.org/index.php/SQL_Injection_Prevention_Cheat_sheet
[3] OWASP Top Ten Project - http://www.owasp.org/index.php/Top_10
[4] OWASP Code Review Guide - http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
[5] OWASP Testing Guide - http://www.owasp.org/index.php/Testing_Guide

[6] OWASP Enterprise Security API (ESAPI) Project - http://www.owasp.org/index.php/ESAPI
[7] Writer's Handbook. MillValley, CA:University Science,1989.
[8] OWASP Legal Project - http://www.owasp.org/index.php/Category:OWASP_Legal_Project
[9] International Journal of Advanced Research in Computer Science and Software Engineering Research Paper Available online at: www.ijarcsse.com
[10] http://buzzupp.com/how-to-secure-your-wordpress-website-from-cross-site-scripting/
[11] http://opensecurity.in/category/white-papers/
[12] https://www.ssllabs.com free online assessment ofpublicfacing server HTTPS configuration

