ISSN No. 0976-5697

Volume 8, No. 5, May-June 2017

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Cyber offences: Emerging Menace combating with spatial tort and Technical countenance with cross site scripting

Madhuri K Shah
M. Tech Department of Cyber Security
Raksha Shakti University
Ahmadabad, Gujarat, India

Chandresh Parekh
Assistant Professor, Electronics and Telecommunication
Raksha Shakti University
Ahmadabad, Gujarat, India

Abstract: This paper explores the vulnerabilities of sites which are vulnerable with cross site scripting in this paper here we have given some example of vulnerable sites with its impact and remedies. these all sites are taken as example if there are sites which have given vulnerability then one can understand the impact, details and remedy of cross site scripting. Developer can save their sites by avoiding this types of loopholes while they are developing sites.it can be helpful to one who want to understand that how cross site actually works and how sites are vulnerable and how websites get vulnerable to the one who have malicious intention..

Keywords: Cross-site Scripting (XSS), DOM-Based Cross site scripting, option method

INTRODUCTION

When we want to check the vulnerabilities of websites for the noble cause to provide more security we first go for vulnerability assessment and cross site scripting is first thing any developer wants to check. Even any hacker or person with malicious intention then he/she will also go first to cross site scripting to steal data.in this paper we see some results of tested websites from that one can get idea that how websites can be vulnerable . XSS (short for Cross-Site Scripting) is a widespread vulnerability that affects many web applications. The danger behind XSS is that it allows an attacker to inject content into a website and modify how it is displayed, forcing a victim's browser to execute the code provided by the attacker while loading the page. Generally XSS vulnerabilities require some type of interaction by the user to trigger the vulnerability, either via social engineering, or waiting for someone to visit a specific page. That's why it's often not taken seriously by developers, but if left unpatched, can be very dangerous. Cross-site Scripting, also known as XSS, is a way of bypassing the SOP concept. Whenever HTML code is generated dynamically, and the user input is not sanitized and is reflected on the page an attacker could insert his own HTML code. The web browser will still show the user's code since it pertains to the website where it is injected.
In such case an attacker can easily insert javascript code which would run under the site's context. By doing so the attacker is able to access other pages on the same domain and can read data like CSRF-Tokens or the set cookies.
If the cookies, which typically contain session identifier information can be read by the javascript, the attacker can use them on his own browser and login to the web application as the victim. If that does not work the attacker can still read private information from the pages, such as read CSRF tokens and make requests on behalf of the user.

HOW CROSS SITE SCRIPTING WORKS

In order to run malicious JavaScript code in a victim's browser, an attacker must first find a way to inject a payload into a web page that the victim visits. Of course, an attacker could use social engineering techniques to convince a user to visit a vulnerable page with an injected JavaScript payload.[10]
In order for an XSS attack to take place the vulnerable website needs to directly include user input in its pages. An attacker can then insert a string that will be used within the web page and treated as code by the victim's browser. The consequences of what an attacker can do with the ability to execute JavaScript on a web page may not immediately stand out, especially since browsers run JavaScript in a very tightly controlled environment and that JavaScript has limited access to the user's operating system and the user's files.[1]
However, when considering that JavaScript has access to the following, it's easier to understand how creative attackers can get with JavaScript.[2]

- Malicious JavaScript has access to all the same objects the rest of the web page has, including access to cookies. Cookies are often used to store session tokens, if an attacker can obtain a user's session cookie, they can impersonate that user.
- JavaScript can read and make arbitrary modifications to the browser's DOM (within the page that JavaScript is running).
- JavaScript in modern browsers can leverage HTML5 APIs such as accessing a user's geo location, webcam, microphone and even the specific files from the user's file system. While most of these APIs require user opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.
- JavaScript can use xmlhttpRequest to send HTTP requests with arbitrary content to arbitrary destinations.
- The attacker injects a payload in the website's database by submitting a vulnerable form with some malicious JavaScript
- The victim requests the web page from the website
- The website serves the victim's browser the page with the attacker's payload as part of the HTML body.
- The victim's browser will execute the malicious script inside the HTML body. In this case it would send the victim's cookie to the attacker's server. The

attacker now simply needs to extract the victim's cookie when the HTTP request arrives to the server, after which the attacker can use the victim's stolen cookie for impersonation.

The above, in combination with social engineering, allow attackers to pull off advanced attacks including cookie theft, key logging, phishing and identity theft. Critically, XSS vulnerabilities provide the perfect ground for attackers to escalate attacks to more serious ones[3]

CLASSIFICATION

Cross-Site Scripting (XSS) attacks occur when[4]

1. Data enters a Web application through an untrusted source, most frequently a web request.
2. The data is included in dynamic content that is sent to a web user without being validated for malicious content.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.[5]

1) Stored and Reflected XSS Attacks

XSS attacks can generally be categorized into two categories: stored and reflected. There is a third, much less well known type of XSS attack called DOM Based XSS i.e.

a) Stored XSS Attacks

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-I XSS.

b) Reflected XSS Attacks

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web site. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

2) Other Types of XSS Vulnerabilities

In addition to Stored and Reflected XSS, another type of XSS, DOM Based XSS was identified the XSS categorization as described: Types of Cross-Site Scripting, which covers all these XSS terms, organizing them into a matrix of Stored vs. Reflected XSS and Server vs. Client XSS, where DOM Based XSS is a subset of Client XSS[6]

SOME EXAMPLE WHICH DESCRIBE SITE VULNERABILITIES

In this paper we tried to elaborate the things which can be seen frequently in cross site scripting. We have taken few websites as testing websites and find its vulnerabilities which can be seen frequently in various XSS cases. We have find vulnerabilities and give its impact with remedies. for finding vulnerabilities we have used tools also[7] table 1 is giving various details for the demo.testfire.net, site and table 2 contain for the testphp.vulnweb.com where table 3 is for scanme.nmap.org.in that we have explained the vulnerabilities, how that particular vulnerability act what are the impact of that vulnerabilities from where that vulnerability can be found. there are various types of vulnerabilities which we have found during testing which are described in details in that tables.one can learn from it that how actual impact is there on site while particular vulnerabilities is there and what possible remedy for that attack. developer can save their sites by going through that list.it may help to save websites from attackers and websites can be maintained in safe mode. That are the result part of testing and thus we can get that what are the deficiency in that sites. To prevent cross-site scripting vulnerabilities it is very important to apply a context dependent output encoding. In some cases it might be enough to encode the HTML special characters, such as opening and closing tags.

In other cases a correctly applied URL encoding is necessary. Links should generally be disallowed if they don't begin with a whitelisted protocol such as http:// or https://, thus preventing the use of URI schemes such as javascript://.Even though most modern web browsers have an inbuilt XSS filter they should not be seen as an alternative to sanitization. They cannot catch all kinds of cross-site scripting attacks and are not strict so not to lead to false positives, which would prevent some pages from loading correctly[8]. A web browser's XSS filter should only be a "second line of defense" and the idea is to minimize the impact of existing vulnerabilities. Developers should not use blacklists as there is a variety of bypasses for them. Another thing they should avoid using is the stripping of dangerous functions and characters as the browsers' XSS filters can't recognize the dangerous payloads when the output is tampered with allowing for possible bypasses. T understand more we have given results and vulnerability Assessment and penetration testing report for all three websites which we have tested.in that tables all details are given of that vulnerabilities which are the available there. Which can make easier to get the understanding of cross site scripting.

Table 1 vulnerability Assessment and penetration testing report

VAPT of demo.testfire.net[9]						
No	Name	Severity	Vulnerability Details	Vulnerability Impact	Affected item	Remedy
1	Cross site scripting (verifi	High	allows an attacker to send malicious code (usually in the form of Java script) to another user. Because a browser cannot know if the script should be	steal the session cookie and take over the account, impersonating the user. It is also possible to modify the content of the page presented to	/comment.aspx /search.aspx	script should filter meta characters from user input

	ed)		trusted or not	the user		
2	DOM - Based Cross site scripting	High	While a traditional cross-site scripting vulnerability occurs on the server-side code, document object model based cross-site scripting is a type of vulnerability which affects the script code in the client's browser	Malicious users may inject JavaScript, VBScript, ActiveX, HTML or Flash into a vulnerable application to fool a user in order to gather data from them. An attacker can steal the session cookie and take over the account, impersonating the user. It is also possible to modify the content of the page presented to the user	/disclaimer.htm	script should filter meta characters from user input.
3	HTML form without CSRF protection	Medium	alert may be a false positive, manual confirmation is required. Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. It found a HTML form with no apparent CSRF protection implemented. Consult details for more information about the affected HTML form.	An attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application	/bank/login.aspx /feedback.aspx /subscribe.aspx	Check if this form requires CSRF protection and implement CSRF countermeasures if necessary
4	User Credentials are send in clear text	Medium	User credentials are transmitted over an unencrypted channel. This information should always be transferred via an encrypted channel (HTTPS) to avoid being intercepted by malicious users	A third party may be able to read the user credentials by intercepting an unencrypted HTTP connection	/bank/login.aspx	Check if this form requires CSRF protection and implement CSRF countermeasures if necessary.

Table 2 vulnerability Assessment and penetration testing report

5	ASP.NET Version Disclosure	Low	The HTTP responses returned by this web application include an header named X-AspNet-Version. The value of this header is used by Visual Studio to determine which version of ASP.NET is in use. It is not necessary for production sites and should be disabled.	The HTTP header may disclose sensitive information. This information can be used to launch further attacks	/	Apply the following changes to the web.config file to prevent ASP.NET version disclosure: <System.Web> <httpRuntime enableVersionHeader="false" /> </System.Web>
6	Click Jacking: X-Frame-Options	Low	The server didn't return an X-Frame-Options header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a	The impact depends on the affected web application	Web Server	Configure your web server to include an X-Frame-Options header. Consult Web references for more information

	Header Missing		page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.			about the possible values for this header
7	Cookie without HttpOnly Flag set	Low	This cookie does not have the HTTPOnly flag set. When a cookie is set with the HTTPOnly flag, it instructs the browser that the cookie can only be accessed by the server and not by client-side scripts. This is an important security protection for session cookies.	-	/	If possible, you should set the HTTPOnly flag for this cookie.

Table 3 vulnerability Assessment and penetration testing report

VAPT of scanme.nmap.org[10]						
No	Name	Severity	Vulnerability details	Vulnerability Impact	Affected item	Remedy
1	Directory listing	high	The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site. Affected items	The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site. Affected items	/images /shared /shared/css /shared/error /shared/error/includes /shared/images/Acunetix /shared/templates	You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.
2	Slow http DOS attack	high	Your web server is vulnerable to Slow HTTP DoS (Denial of Service) attacks. Slowloris and Slow HTTP POST DoS attacks rely on the fact that the HTTP protocol, by design, requires requests to be completely received by the server before they are processed. If an HTTP request is not complete, or if the transfer rate is very low, the server keeps its resources busy waiting for the rest of the data. If the server keeps too many resources busy, this creates a denial of service. Affected items	Web server	A single machine can take down another machine's web server with minimal bandwidth and side effects on unrelated services and ports.	Consult Web references for information about protecting your web server against this type of attack
3	Broken links	medium	A broken link refers to any link that should take you to a document, image or webpage, that actually results in an error. This page was linked from the website but it is inaccessible. Affected items	/advertising.html /fyodor /privacy.html	Problems navigating the site.	Remove the links to this file or make it accessible.
4	Email found	medium	One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and	/shared/webabuse.html	Email addresses posted on web sites may attract spam.	Check references for details on how to solve this problem

			email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found. Affected items			
5	Option method is enabled	medium	HTTP OPTIONS method is enabled on this web server. The OPTIONS method provides a list of the methods that are supported by the web server, it represents a request for information about the communication options available on the request/response chain identified by the Request-URI. Affected items	Web Server	The OPTIONS method may expose sensitive information that may help an malicious user to prepare more advanced attacks	It's recommended to disable OPTIONS Method on the web server

REFERENCES

[1] Wassermann and Z.su "statistic Detection of security in scriptting Vulnerabilities", In ICSE, 2008

[2] Weinberger, P.Saxena, D Akhawe, M Finifter, R. AAA, and D. Song "A Systematic Analysis of XSS sanitization in web Application Framework", In ESORICS,2011

[3] http://www.ijcns.com/pdf/ijpcscvol4no22012-1.pdf

[4] Jayamsakthi Shanmugam, Dr.M.Ponnavaikko, "Behaviour-Based Anomaly Detection On the server side to Reduce the effectiveness of cross site Scripting Vulnerabilities", in preceedings of 3rd IEEE International conference on semantics knowledge, and Grid, Published by IEEE computer society in IEEE Xplore, China , pp. 350-353, october 29-31 2007

[5] https://www.owasp.org/index.php/OWASP_Categories

[6] https://www.owasp.org/index.php/OWASP_Categories

[7] https://owasp.org/index.php

[8] http://acunetix.com/website

[9] http://demo.testfire.net/

[10] http://scanme.nmap.org.