Volume 5, Issue 2, February 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey of XSS Payload & Exploiting Issues of XSS Vulnerabilities

Tejinder Singh Mehta*, Sanjay Jamwal
Department of Computer Science
Baba Ghulam Shah Badshah University,
Rajouri, J&K, India

Abstract— The attacks are easy to make but tough to be mitigate. As Social Sites get more and more users across Internet, Cross Site Scripting is becoming one of the major problems, which results in serious consequences, such as theft of some personal trusted data and information. This paper discusses the XSS payload and extensive issues which were used by an attacker to apply them on client machines or server for making malicious web pages like XSS or SQL Injection.

Keywords— XSS: Cross Site Scripting; HTML: Hyper Text Markup language; URL: Uniform Resource Locator; SQL: Structured Query Language; WAS: Web Application Scanning

I. INTRODUCTION

XSS worms that exploit a security vulnerability known as cross site scripting within a web application, infecting users in a number of ways depending on the vulnerability type. Such type of features as profiles and chat systems can be affected by XSS worm's vulnerability, when implemented improperly or without regard to security. Often, these worms are specific to single web applications, spreading widely by exploiting specific vulnerabilities. Cross-site scripting vulnerabilities are commonly exploited in the form of worms on popular social sites or commercial web applications, such as MySpace, Yahoo!, Orkut, Justin.tv, Facebookand Twitter. These worms can be used for malicious intent, giving an attacker the basis to steal personal information provided to the web site, such as passwords or credit card numbers.

A. XSS Payload:

XSS payloads in which we locate the vulnerabilities lie within the execution of application. Such attacks can directly attack the websites with which they can affect the XSS vulnerability. Even Same Origin Policy can't control all client side application from running JavaScript.

B. Session Payload:

Session theft payload directly affect the website because of the malicious code injected to exploits the client system with XSS so that attacker can get the access of same controls to web applications, because of the current client's web browser state of authentication the attacker can easily exploit by creating HTTP request. Even Same Origin Policy can't fully prevent this script of malicious contents. So http response can be easily read it to find which action is to be done to target that particular http requests.

Definition: - Session Payload or Session hijacking is an arbitrarily action to change the state of authentication. So the hacker or an attacker can put his malicious code to access the system of client and makes it vulnerable.

From the definition of session hijacking empowers the attacker to exploit the web applications and session hijacking requires a real time interaction for it's automatically execution [10]. In fact session's theft or session attacks are further classified into two classes:

1.) ID Theft Payload:

In web application session identifiers (SID) is used to track the state of authentication for different users. Every user had SID and it belongs to authenticated user. In this category what an attacker do reading SID value with JavaScript and redirected to attacker location. Also, a subset of modern web browsers provides "httponly" cookies which allow SID storage which is not accessible by JavaScript [11]. So he can get the credential details with this sessions ID theft can be done by Impersonating Session Identifiers.

<Script>
Document.write("img height = 0, length =0 width =1 src =http ://victim. Org ?SID = + document. cookie +">);
</Script>

In recently client side SSL or http authentication the attacker fails to attack if non SID based authentication used.

2.) Browser Payload:

Session hijacking never needs session identifier (SID) over the internet. The attack is possible only with victim browser. Modern browsers provide XMLHTTPRequest Object use GET and POST methods to request the URL. That

follows Same Origin Policy hijacking is totally different instead of SID. Browser hijacking is very much effective because the attacker put the series of http request to the web server. Even web server fails to differentiate which is the regular request initiated from user or the request is processed by script. Thus malicious JavaScript got the capabilities to run automatically and causing much sever problems of injection.

In 2005 worm vulnerability employed the same techniques for self replicating JavaScript and infected million of clients on myspace.com and famous with the name of "samy is my hero"

This attack is not possible with XMLHTTPRequest object by hidden frames. It can also be executed using hidden iframes [12].

3.) Acquirement to XSS Proliferation Payload:

Not all attacks on web application are vulnerable to XSS. It is possible in above discussion because the user visits one vulnerable page and it's written in JavaScript. For example when credit card is to be used for shopping its information is stored in the web browser. In order to steal such information a malicious script is to run on client machine. Suppose one web page is XSS vulnerable and other one is not. To steal such credentials attacker make a propagation of page A to page B. There are two scenarios as discuss below:

- **Propagation via iframe inclusion:**

XSS can replace the web page with an iframe further attacking script needs iframe to display an attacked web page. As long the user doesn't leave the application domain the malicious code is able to witness the user surfing to a web page that is used to display iframe.

- **Propagation via pop under window:**

Second way is to pop under the windows. In certain time on fast window the other browser window pop up the second window of browser. The malicious script opens up new window of the browser

C. Password Theft Payload:

Instead of hijacking sessions it is better to hijack the user password. User stores simple and easy passwords for multiple sites. The better way is to target a web application that makes the threats more potential and strong enough by cracking a password. The user's password., as users often reuse the same password for multiple sites [13]

1.) Applications Authentication Dialogue:

If the malicious script is injected into a web page in which the original authentication of user credentials is also displayed. Then attacker has many ways to obtain the user or victim password very easily.

Reading the password with the JavaScript

From a technical view html password values are nothing but ordinary text fields. So it is not a good thing to use JavaScript that can make web pages easily vulnerable.

```
<script>
function leakData(){

  var data = document.getElementById('username').value +
             ':' + document.getElementById('password').value;
  var url = 'http://attacker.org/log?data=' + escape(data);
  var img = document.createElement('img');
  img.setAttribute('src', url);
  img.style.height = '0';
  img.style.width = '0';
  document.getElementsByTagName('body')[0].appendChild(img);

  /*
  ** Delay the submit() for so that the leaking HTTP request succeeds reliably
  */
  window.setTimeout(function(){
    document.getElementById('loginform').submit();
  }, 100);

  return false;
}

document.getElementById('loginform').onsubmit = leakData;
</script>
```

Fig 1: Password Cracking Technique [7]

Rerouting the authentication request:

Man in middle attack is more dangerous than Session hijacking and password theft technique to access the credentials of user. We had a situation i.e. man in the middle to alter the authentication form. When the user submit the authentication form resulting http request, having information about the password or credentials details were to be displayed with the URL. So the user login the data attacker is able to redirect the data by displaying error 404 towards the client response. If SSL is used in authentication form and attacker with the help of https response URL is tried to be manipulated it will revert the message that connection is not secure to the web browser.

2.) Abusing the Password Manger:

Modern browser provides us password manager functionality and sometimes browser tells us to remember the passwords for further use. So the password manager stores the credentials details. So when user requests for the password the authentication form fills the details of username and password. Such behavior helps to exploit Like SQL Injection or

XSS. There are two ways to exploits by XSS.Firstly the attacker sends malicious code to authentication page as we discussed in passwords cracking with the JavaScript and after that browser manager fills the credentials details on his behalf.

3.) Spoofing Authentication Forms:

Finally the attacker can create authentication form instead of original form. By using DOM tree, the attacker attacks the html and spoofed into a websites very easily and returns a spoofed message to client system with which hacker gets the credentials details. Several techniques to spoof authentication forms are discussed in [7].

D. EXPLOITING XSS ISSUES:

This issue introduced the concept of offensive capabilities of JavaScript. First we discuss the JavaScript driven attack which initiated attacks with legal approaches of JavaScript malicious code executed within web browser. Second we define the XSS pay which implies attack and run through XSS exploit and figure out previous policy fail in such cases. Third case introduces basis mechanism which makes the module of XSS payloads and in last section we give systematic classifications of XSS payloads.

1.) JavaScript Driven Attacks (JSDAs):

JavaScript helps the programmer with robust capabilities for comparing client side code. But on the other hand JavaScript also helps the attacker with a lot of offensive techniques. When user visits the web page having such malicious script then it exposes the JavaScript driven attacks. The most property of JavaScript driven attacks is that attacks employ legitimate meaning which properly describes language specification. This distinguishes this class of attacks from browser based exploits which rely on security vulnerabilities within the browser's source code [14].

2.) Defensive Technique:

There are two techniques which help the user to avoid the JSDA attack.

- Visit the web sites which are secured to user but we can't take out malicious actions.
- Deactivate the JavaScript or disable the JavaScript in order to protect the user from attacks.

These two techniques combine with an effective policy:

Only allow JavaScript for explicitly trusted web applications for the web browsers. With the birth of the "Web 2.0" phenomena [15] and the trend towards rich web applications which mimic the behaviour of their desktop counterparts, the usage of JavaScript became ubiquitous and in most cases mandatory

3.) Loophole in Same Origin Policy:

JavaScript dynamically include elements from DOM tree. Same Origin Policy applies on a document level creates a loophole in SOP:

- The script creates HTTP request of URLs to external locations by including the items to DOM.
- The Script can intercept events which are triggered by exclusive process.
- When the inclusion process is completed. The remote items is part of same document as the script.

Cross Site Scripting is one of the vulnerability found in web applications. Cross Site Scripting enables attacker to inject client side scripting into the web pages viewed by other users. In Cross Site Scripting, attackers use this vulnerability to bypass access controls and nearly 84% of all vulnerabilities are documented by XSS. In Cross Site Scripting an attacker can gain access to sensitive data in web page contents, such as session cookies and a variety of information maintained by the browser. Cross Site Scripting surpassed all vulnerabilities with researchers in 2007 as many 68% of websites most likely open to XSS attacks. Cross Site Scripting problem is because of website's server code failure and HTML contents and vulnerability like SQL injection sometimes do occur with XSS.

Cross Site Scripting vulnerability allows attackers to execute commands and display arbitrary contents in a victim user's browser. Cross Site Scripting vulnerability is web based applications in which attackers attack sensitive information, such as stealing of Cookies and Sessions IDS, reflects the malicious code to victim users, Deface or hijack attack and provide entry point for large scale attack against organization assets and user data.

Few challenges like Cross Site Scripting and Sql Injection which are difficult to mitigate.

- Injection by URL.
- Injection by exploiting client side code.
- Injection via permanently displayed data.

II. LITERATURE REVIEW

A. Vulnerability Analysis:

Vulnerability analysis, also known as vulnerability assessment, is a process that identifies the security holes (vulnerabilities) in a computer network. Vulnerability analysis consists of several steps:

- Defining and classifying network or system resources
- Identifying potential threats to each resource
- Developing a strategy to deal with the most serious potential problems first
- Defining and implementing ways to minimize the consequences if an attack occurs.

If security holes are found as a result of vulnerability analysis, a vulnerability disclosure may be required. If the vulnerability is not classified as a high level threat, the vendor may be given a certain amount of time to fix the problem before the vulnerability is disclosed publicly.

The stage of vulnerability analysis (identifying potential threats) is sometimes performed by a white hat using ethical hacking techniques. Using this method to assess vulnerabilities, security experts deliberately probe a network or system to discover its weaknesses.

B. Server-Side:

With server-side scripting, completing an activity involves sending information to another computer (server) across the internet. The server then runs a program that processes the information and returns the results, typically a webpage. Search engines use server-side processing. When a keyword is sent, a program on a server matches the word or phrase entered against an index of website content. (To complete the same search as a client-side process would require the browser to download the entire search engine program and index.) Server-side scripting languages include ASP and PHP.

C. Filtering JavaScript:

Protection from malicious JavaScript code can be achieved at different locations. First users can protect themselves by disabling JavaScript (or active content in general) in their browsers. Unfortunately this renders most of the modern Web sites unusable. Therefore Web applications must protect their users by filtering out malicious JavaScript.

XSS attack, JavaScript Malware was considered as serious threats to browser security. Just a simple click and you could get in hacked list and even (SSL) secure socket layer breach of security like heart bleed bug problem shook the whole world that even HTTPS,SSL is not secure. Firewall, antivirus, anti phishing, 2Dl authentication or any other tool couldn't save them from occurring of such attacks.

Joon , Ravi, 2000 [2] describes the testing tactics and provide mechanism for preventing website from vulnerabilities like XSS, SQL Injection attacks. If any other vulnerability exists in server than it would be very much difficult to protect it on client side system. Further Kruegeret al Christopher, Vigna, Robertson, 2005 [1] describes that it would be difficult to protect from IDS [2] Joon , Ravi, 2000. CERT (center of internet expertise) as per their views no client side solution can be completely safe and authentic Yan wen, Yu Christian Hang, Chunh Tsai,2004 [3]. Few ideas of research Jovanovic , Kruegel and Kirda, 2006 [4] Wes Masin and Andy Podgurshi stated that, information flow work will increase false positive rate. Some validation mechanism Chung Hung,Chung hang Tsai, 2007 [6] and scanners pose to prevent XSS vulnerability. Detection techniques can accurately sensitized by false negative and false positive rate. False positive rates among total alerts of vulnerabilities and David Scott Scott D. Sharp, 2002 [5] suggested policies for input validation and requires correct validating policies for entry point in a web site.

III. CONCLUSION AND FUTURE WORK

For securing web application, several researches have been conducted in three areas.

- Defensive coding with new language and security system.
- Monitoring systems and prevention at run.
- Vulnerability detection.

First approach for defensive systems with a new language must be created with all authentic parameters. Second approach is for monitoring systems and preventing at runtimes, implemented with firewall or browsers plug-in. Third approach vulnerability detection is conducted manually or by developers or via use of vulnerability scanners.

From the survey of different articles/research papers it is found that Cross Site Scripting are the most powerful and easiest way to attack on web sites. This research work will help us to make web pages less vulnerable. Our review finds that there are problems in existing techniques because the developers barely pay attention to the security of websites and web browsers while developing website. We will work on QualysGuard WAS and will design a model to prevent website from XSS vulnerability.

REFERENCES

[1] Christopher Krugel G Vigna William Robertson, 2005 " A Multi Model Approach to Detection of Web Based Attacks", Computer Networks, Volume 48, Issue 5, pp 717-738, August 2005.

[2] Joon S Park, Ravi Sandh,2000 "Secure Cookies on the web ",IEEE internet computing . Volume 4. Pp. 36-44 July/August 2000.

[3] Yan wen Huang Fang Yu Christian Hang, Chunh Tsai Der Tsal Lee Sy Yen Kuo,2004" Securing Web Applications Code By Statics Analysis and Routine Protection ", In Proceedings of International WWW Conference New York Usa, pp 40-52.

[4] N. Jovanovic ,C Kruegel and E Kirda,2006 " Pixy: A Statics Analysis Tool for Detecting web Applications Security Vulnerabilities ", In Proceedings of 2006 IEEE Symposium on Security and Privacy (S&P'06),California, USA pp. 27-36

[5] Scott D. Sharp,2002 "Abstracting Application Level Web Security" In Proceedings of 11th International Conference world Wide Web (WWW2002). Honolulu Hawaii. Pp 396-407.

[6] Yao wen Chung Hung Tsung Po lin and Chung hang Tsai,2007, "Web Application Security Assessment By Fault Injection and Behavior Monitoring", In Proceedings of 12th international conference on World Wide Web, Budapest, Hungary. Pp 861-999.

[7] Mieke Hildebrandt. Web authentication revisited. Master's thesis, University of Hamburg, June 2008.

[8] [Mar,25 2015] acunetix website.[online] Available http://www.acunetix.com/blog/articles/non-persistent-xss

[9] [Jan, 2014]neokobowebsite.[online]Available:http://neokobo.blogspot.in/2014/01/3218-client-side-attacks. html
[10] Benjamin Livshits and Weidong Cui. Spectator: Detection and Containment of JavaScript Worms.In Usenix Annual Technical Conference, June 2008.
[11] MSDN. Mitigating Cross-site Scripting With HTTP-only Cookies. [online], http: //msdn. microsoft.com /workshop/ author/dhtml/httponly cookies.asp,(01/23/06).
[12] JulienLamarre.AJAXwithoutXMLHttpRequest,frame,iframe,JavaorFlash.[online],http://zingzoom.com/ajax/ajax with image.php, (02/02/2006),September 2005.
[13] Roger A. Grimes. MySpace password exploit: Crunching the numbers (and letters).[online], http:// www.infoworld.com/article/06/11/17/47OPsecadvise1.html (07/01/08), November 2007.
[14] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In USENIX Workshop on Hot Topics in Understanding Botnets, April 2007.
[15] Tim O'Reilly. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications & Strategies, (65), 1. quarter 2007.

