PREVENTING CSRF ATTACKS BY VERIFYING REDIRECTION REQUEST AND USER SESSION

Dr. Purva N. Desai.

Abstract - These days Internet has become handy and most advanced useful technology due to use of various electronic gadgets. Various online services provided by Internet helps the present human civilization to such a greater extend that life without internet seems to be impossible. Due to its omnipresence, Internet has started attracting hackers/attackers who keep looking for new techniques to create maliciousness in web application. According to researchers and industry experts, the Cross-Site Scripting (XSS) is the one of the top most vulnerability found in the web application. Here, injected malicious code executes on the browser's site which affects victims badly. This paper focuses on Cross-Site Scripting Redirection and Cross-Site Request Forgery attacks which is sub categories of XSS attacks. This paper further discusses the proposed algorithm which protects web application against such redirection attacks by verifying URL request made by user and their session.

Keywords: XSS, CSRF/XSRF, CSSR/XSSR, one-click attack, session riding attack

I. INTRODUCTION

In this current age of information and technology, most of the tech-savvy population of the world became heavily dependent on the Internet. Various services provided by Internet has changed present civilization to climb its new and wider dimensions. It is bitter truth that without Internet our day-to-day life seems to be impossible.

Today thousands of web sites including customized services have become key part of most of the human activities. But on the other side unfortunately, it has attracted hackers or attackers too who always keep eyes and tries to create new techniques to sabotage fruitful-skilled advance technology by intruding in to web application. XSS is one of the top most vulnerability in the web application.

XSS is one kind of application layer web attack in which they try to inject malicious scripts to perform malicious actions on any trusted web sites to fulfill only their nominal or bigger self-interest. According to Shalini & Usha (2011) [1], it is called "cross-site" because it involves interaction between two web sites to achieve attacker's goal. In XSS, malicious code executes on web browser side which badly affects users.

In normal case, XSS executes when the web page is loaded or associated event occurs. XSS is not only embedded in JavaScript and HTML, but also in VBScript, ActiveX, AJAX, action script like flash or any other browser executable scripting language and mark-up language.

For many reasons XSS can be used such as Take over user' account, Spread worms, Trojan horse, Control access of browser, Phishing, Expose of the user's session cookie, Redirect the user to some other page or site, Modify presentation of content, Bypass restrictions, Malware attacks & DoS attack, Fake advertisement, Click fraud, etc.

II. SUB CATEGORIES OF XSS ATTACKS

Followings are the sub categories of XSS attacks which are used to redirect a particular victim unwillingly.

a. **XSSR or CSSR:** XSSR or CSSR stands for Cross Site Script Redirection is used to redirect a victim to another page unwillingly for example mouse over event redirect user to some malicious page. The page can for example contain a phishing template, browser attack code or in some cases where the data or JavaScript URI scheme is used for session- hijacking.

b. **XSRF or CSRF:** XSRF or CSRF (sometimes referred to as C-Surf) stands for Cross Site Request Forgery which is used to send automated input via the user to the target site. XSRF can in some cases be triggered just by viewing a specially crafted image tag. CSRF is also known as one-click attack or session riding. According to Kombade and Meshram (2012) [2], CSRF can be stored CSRF or Reflected CSRF.

Here, in both case, victim is redirected to some malicious web page unwillingly. These attacks are form as either stored redirection attacks or reflected redirection attacks. In stored redirection attacks, attacker injects exploitable links or other contents in web application itself. In reflected redirection attacks, attacker sends exploitable links or contents to victim via message post, e-mails, blog or instant messages.

Stored redirection attacks are more fruitful for attacker because the user who receives such exploitable links or contents is at present genuine user that performs

web actions undoubtedly. Whereas reflected redirection attacks may get fail frequently, as users may not be currently active into the target system where exploitable links and contents are available. This proposed methodology only protects against stored redirection attacks. It does not protects against reflected redirection attacks because source of attack is not fixed i.e. attack can be delivered via e-mails, chat room, blog, etc.

III. REVIEW OF LITERATURE

Jovanovic et al. (2006) [3] proposed mitigation technique for cross-site request forgery. Here authors converts GET request into POST request. Authors also checks referrer header of HTTP request to identify request forgery. This technique works as server site proxy.

Zeller and Felten (2008) [4] creates plug-in for Firefox web browser. This plug-in checks every HTTP request. If request is made using GET method or if request contains same origin reference or request contains allowable cross-domains then these requests are considered as valid request and are allowed to perform further processing. Other than these valid requests are rejected by giving alert to user.

Ramarao et al. (2009) [5] authors present client-side solution which is proxy that used to identify CSRF attacks by observing IMG elements. Here, authors mitigate CSRF attacks by identifying image extensions in static image URL. It disables GET request which is automatically dispatched. If proxy founds invalid URL, it disables image in web page. If web page contains dynamic image, it prompts user whether to allow it or not. This method does not check JavaScript img objects. Attack vector is not detected here if it is injected in longdesc attribute of IMG tag.

Alexenko et al. (2010) [6] come with idea to protect history of user's preference. For this, they developed extension of Firefox browser. Here user can add their request URL as referrer fields. This URL list can be matched with HTML tags to detect request forgery attack using XSS. If some unknown links are found then system informs about it to user that whether he/she wants to access this URL. If user is agreed to access URL then this URL is added to list.

Chen et al. (2011) [7] presented study of CSRF guard. Authors describe working model of CSRF guard, model that bypass CSRF guard, its limitations and various security scenarios. Here, authors suggest that security must be implemented at web site development life cycle.

Gupta and Sharma (2012) [8] introduced a sandbox environment which protects cookies against XSS attacks. This environment set at client-side. It prevents unknown threats that make change in program or data. Its only protect against changes, one can make attempt to read cookies value stored in client computer.

Sankuru and Janjanam (2013) [9] proposed approach works on reflected CSRF attack at client site. In this approach, unique CSRF token is appended with valid request and this token is set as part of session data. If web site request contains this unique CSRF token, it process at server site otherwise request will be rejected.

Sentamilselvan et al. (2014) [10] proposed algorithm for stored attacks and login CSRF attacks. To prevent stored attacks, pattern recognition method is used. To prevent login CSRF, 2-step verification method is used. For this verification, random number sent to user via mail or SMS by server. Using this random number, one can view his/her home page.

Kavitha and Ravikumar (2015) [11] proposed algorithm which protects against clickjacking. This algorithm follows regex approach. In this proposed architecture, authors checks all iframe available in a web page then it also checks IP address as well as domain. If user enters URL, it checks against iframe URL using regex. System also maintains list of DNS as DNSloopup. If more than 10 requests found as bad request, this user IP is blocked.

Agrawall et al. (2017) [12] analyzed working of FIM protocol and how it vulnerable to cross origin attacks. In this paper, authors also verified that various cross origin attacks can be mitigated using CORP. Further this paper carried out experiment to show the vulnerabilities present in the authentication endpoints: Login and Logout, and mitigate the risks using CORP. The generic model introduced in this paper subsumes many specific cross-origin attacks, such as CSRF, clickjacking, cross-site timing attack, login detection, and validates the soundness of CORP in mitigating these attacks.

Semastin et al. (2018) [13] described CSRF HTML file, proxy listening, auto submission of forms and creation of form in the browser criteria for comparison of the CSRF testing. This paper also specifies various measures like: prevention against GET and POST requests, single step transactions, usage of random value in the code, URL and cookies, random value encryption, browser and domain dependency of mechanism for different solutions. This solution provides protection to the application against CSRF attacks even if the attacker gets access to the token in the URL, the attack will not be executed successfully because of the double validation.

Srokosz et al. (2018) [14] introduced a new web-based architecture for protecting web applications against CSRF attacks. They extend a classic, static WAF approach with historical and behavioral analysis, based on actions performed by the user in the past which reduces the need for additional forms of authorization. Here, after collecting and analyzing user's history, the additional authorization appears only in the situation that actually requires it. Such approach increases the responsiveness and general feel of the application.

Parimala et al. (2018) [15] proposed CSRF vulnerability detection tool using python. This is the scanning tool which take URL of the website as and input, to test for vulnerabilities as per the listed vulnerability. For a valid URL, it will crawl the Website with the help of their own developed Crawler. The Crawler will parse the website. Scanner will test all the forms and links of the website for the vulnerabilities. Authors takes token, CSRFTOKEN, authenticity token, csrfmiddlewaretoken.as parameters for vulnerability detection.

IV. Research Objectives

The main objectives of this research paper are as follow:

- Identifying aspects of request forgery attacks.
- Protect web application from such redirection attacks.

V. Methodology

This methodology is divided into two parts: part 1 for simple redirection and part 2 for request forgery. In simple redirection, somehow attacker succeed to add malicious URL reference in given web page by using XSS. Here, victim is automatically redirected to vulnerable site unwillingly by referring this malicious URL. This attack is also known as one-click attack.

In request forgery, if victim is logged into some genuine web site then session is created to continue transactions. Here, if victim opens new tab in browser and tricks into attacker's website, a script is automatically executed which fetch victim's session id. Using this cookie/session id, attacker performs transactions through legitimate web site as if he is authorized user. This type of attack is also known as session riding attack. Figure 1 shows scenario of request forgery attack: To protect against simple redirection attacks (part 1) and request forgery attacks (part 2), following procedure is required to be followed: Here, proposed algorithms can be implemented at web development phase. For part 2 attacks, when user

provides login details in given web page, web developer need to set and verify user session for authorized user as mentioned below to protect against attack.

Algorithm for XSSR & XSRF prevention methodology: Following section defines working flow of protection module that works against **part 1** attack as algorithm and flowchart.

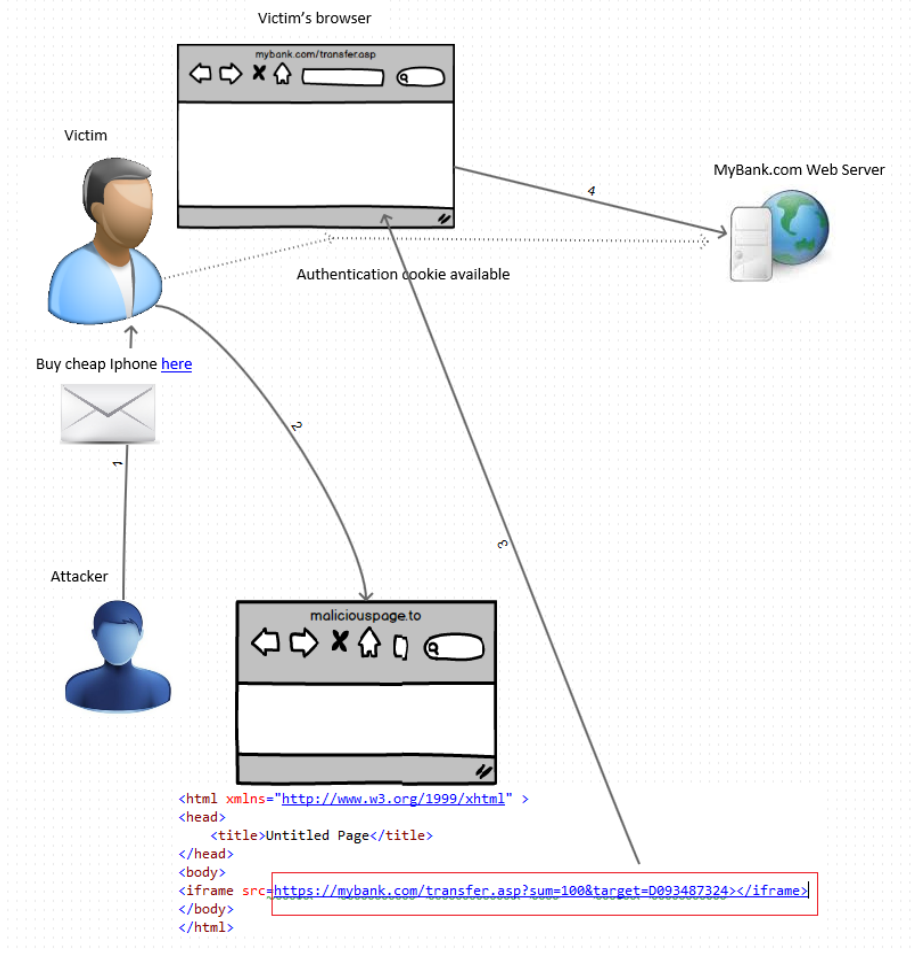

Fig. (1) Scenario of request forgery attack

Here, SiteURLTable is used that contains list of URL supported by given web site.

Step 1: Take a web page as input. Fetch all the element tags of given web page.

Step 2: Check whether element tags contains 'href' or 'src' attribute or not. If such attributes are not available then continue web page processing. If 'href' or 'src' attribute is identified then fetch its attribute value for further checking.

Step 3: Compare attribute value with data entries of SiteURLTable. If value match found then set isAttackFound variable to false and allow further web processing.

Step 4: If its attribute value does not match with SiteURLTable data entries then set isAttackFound variable to true and destroy given web request.

Following section defines working of protection module against part 2 attacks. This algorithm helps web developer to protect session and cookies against request forgery attacks that leaks sensitive information of victim. To do so, web developer needs to create protected session. Web developer also needs to check certain information with every web page request. Session id needs to be changed repeatedly.

Step 1: When web user provides login details (i.e. user id and password) to access web pages, verifies it with user value stored in data table of database to authenticate given user.

Step 2: If login details provided by web user is matched with details of data table, consider user as authorized user and initiate new session for this user.

Step 3: Store user id, browser details (like name of browser and browser version) and IP address as session variables. Encrypt these values of user session.

Step 4: Maintain pageLoadCnt variable as counter variable and initialize it by zero for each page. Increment this variable by one when any web page is requested which is only accesible to registered user.

Step 5: Check value of pageLoadCnt variable for each page request which is used by only registered user. If value of pageLoadCnt variable is

greater than 3 then recreate current session id and reset counter variable to zero.

Step 6: Fetch browser details and IP address of web user, for each page being request and stored it in temp variables.

Steps 7: Compare value of temp variables with value of session variables. If details are matched, set isAttackFound variable to false. Otherwise set isAttackFound variable to true.

Step 8: If isAttackFound variable set to true, web page request is consider as request from fake user, destroy current session values.

Step 9: If isAttackFound variable set to false, web page request is consider as request from authorized user and allow accessing of web page.

VI. RESULT AND DISCUSSION

The methodology described is implemented and results are obtained for the purpose of data analysis and interpretation. The vulnerability performance test (VPT) is conducted using 120 sites randomly selected which are highly ranked commercial sites based on the popularity index of various search rank index.

Table 1: Fold 1 observation without implementing the algorithm

Sr.No.	Pattern Type	Attack Fired	
		Nos.	Detected
1	XSS Attack	82	14
a)	CSS Based	37	7
b)	HTML Based	19	7
c)	JavaScript parsing based	14	0
d)	Others	12	0
2	XSS using HTML quote Encapsulation	24	0
3	URL String Avoidance	14	0

This VPT test is formed in two folds: (1) without implementing the algorithm and (2) by implementing algorithm. During this test, the listed sites were checked for different types of CSRF pattern attacks on these two folds. The results are obtained as shown in Table 1, 2 and 3:

Table 2: Fold 2 observation by implementing the algorithm

Sr.No.	Pattern Type	Attack Fired	
		Nos.	Detected
1	XSS Attack	82	80
a)	CSS Based	37	36
b)	HTML Based	19	19
c)	JavaScript parsing based	14	13
d)	Others	12	12
2	XSS using HTML quote Encapsulation	24	22
3	URL String Avoidance	14	13

Table 3: Consolidate Vulnerability observation

Sr.No.	Pattern Type	% Detection Rate	% Vulnerability
1	XSS Attack	97.56	2.44
a)	CSS Based	97.30	2.70
b)	HTML Based	100.00	0.00
c)	JavaScript parsing based	92.86	7.14
d)	Others	100.00	0.00
2	XSS using HTML quote Encapsulation	91.67	8.33
3	URL String Avoidance	92.86	7.14

It is observed that 76.67% sites were affected by redirection attacks and 73.33% sites were affected by request forgery attacks. From the observations it can be concluded that exceptionally high detection rate is observed in case of HTML based scripts and other types of Scripts attacks which is 100% level of accuracy when the

algorithm is implemented. It is also important to note that above 95% of detection rate is observed in case of CSS based script attacks. Overall above 97% of accuracy level of detection observed in case of all types of XSS based Script attacks. By implementing proposed methodology, accuracy protection level against simple redirection and request forgery type is increased to be 97.78% and 96.67% respectively. The accuracy obtained is highly significant. This methodology improves detection rate from 96.67% to 100%.

VII. CONCLUSION

Web sites are introduced to provide worldwide connectivity, to provide information and services, to make user's work easier, to save transaction time and human efforts. World Wide Web (WWW) is one of the valuable resources which provides communication channel to the entire globe. So no one has right to misuse it for personal gain or for harassment. Cross-Site Scripting is the simplest way for an attacker to gain user's confidential information. This paper presents technique that identifies as well as prevents redirection attacks by monitoring URL request and verifying session of user.

VIII. REFERENCES

[1] Shalini, S., & Usha, S. (July, 2011). Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side. IJCSI International Journal of Computer Science, Vol. 8(4), (pp. 650-654).

[2] Kombade, R. D., & Meshram, D. B. (February 2012). CSRF Vulnerabilities and Defensive Techniques. I. J. Computer Network and Information Security, 1, (pp. 31-37).

[3] Jovanovic, N., Kirda, E., & Kruegel, C. (2006). Perventing cross site request forgery attacks. International Conference on Security and Privacy in Communication Networks (SecureComm). IEEE.

[4] Zeller, W., & Felten, E. W. (2008). Cross-Site Request Forgeries: Exploitation and Prevention. Technical Report, Princeton University.

[5] Ramarao, R., Radhesh, M., & Pais, A. R. (Aug 2009). Preventing Image based Cross Site Request Forgery Attacks. Indo-US conference and Workshop on Cyber Security, Cyber Crime and Cyber Forensics. Information Security Research Labs.

[6] Alexenko, T., Jenne, M., Roy, S. D., & Zeng, W. (2010). Cross-Site Request Forgery: Attack and Defense. 7th IEEE Consumer Communications and Networking Conference (CCNC). Las Vegas, NV: IEEE.

[7] Chen, B., Zavarsky, P., Ruhl, R., & Lindskog, D. (2011). A Study of the Effectiveness of CSRF Guard. IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing, (pp. 1269-1272).

[8] Gupta, S., & Sharma, L. (Dec., 2012). Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense. International Journal of Computer Applications, Vol. 60(14), (pp. 28-33).

[9] Sankuru, R., & Janjanam, M. (Aug, 2013). Web Application Security-Cross-Site Request Forgery Attacks. International Journal of Computer Science & Engineering Technology (IJCSET), Vol. 4(8). (pp. 1194-1200).

[10] Sentamilselvan, K., Lakshmana Pandian, S., & Ramkumar, N. (November 2014). Cross Site Request Forgery: Preventive Measures. International Journal of Computer Applications, 106(11), (pp. 20-25).

[11] Kavitha, D., & Ravikumar, S. (June, 2015). Enhanced Vulnerability Analysis For Clickjacking Web Attack And Providing Security Using Whitelisting URL Analyzer. International Journal of Engineering and Computer Science (IJECS), 4(6), (pp. 12652-12657).

[12] Akash Agrawall, Maheshwari Shubh Jagmohan, Projit Bandyopadhyay, Venkatesh Choppella. (Dec., 2017). Modelling and Mitigation of Cross-Origin Request Attacks on Federated Identity Management Using Cross Origin Request Policy. 13th International Conference on Information System Security. Report No: IIIT/TR/2017/-1

[13] Emil Semastin, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, Mirjam Jonokman, Ganthan Narayana Samy, Sundresan Perumal. (Jan., 2018). Preventive Measures for Cross Site Request Forgery Attacks on Web-based Applications. International Journal of Engineering & Technology. 7 (4.15). (pp. 130-134)

[14] Michal Srokosz, Damian Rusinek, Bogdan Ksiezopolski (Sep., 2018). A new WAF-based architecture for protecting web applications against CSRF attacks in malicious environment. Proceedings of the Federated Conference on Computer Science and Information Systems. Vol. 15. (pp. 391–395).

[15] Parimala G, Sangeetha M, R Andal Priyadharsini. (2018). Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery. National Conference on Mathematical Techniques and its Applications.

AUTHOR'S PROFILE

Dr. Purva N. Desai obtained her Ph.D. in area of Web Security from the JJTU, Jhunjhunu, Rajasthan and M.Sc. (ICT) from the VNSGU, Surat, Gujarat. She is serving at Vivekanand College for Advanced Computer and Information Science as an Assistant Professor since 2008. She has presented many research papers at State level, National level and International level conferences. She has achieved Best Research Paper award for "Detecting Cross-Site Scripting Attack Based on Input Analysis" at International Conference on Emergence of India as a Global Economy: Challenges and Opportunities", Sterling Institute, Mumbai. She is also author of the books entitled "E-Commerce & Cyber Security" published in year 2015 and "Essence of Operating System" published in year 2018. She got four time 100% result award in subject of "E-Commerce and Cyber Security".