

Understanding the Cross-site Request Forgery Vulnerability (CSRF) and Detecting CSRF vulnerability using Python Script

Kherala Ankita A

¹M.Tech, School of Information Technology & Cyber Security, Raksha Shakti University

Abstract- In today's world web applications are very important for our daily routine and many other activities are relay on the security of these web applications. Due to lack of security most of web applications are still vulnerable with the CSRF vulnerability. Cross-site request forgery (CSRF/XSRF) is vulnerability that is found in most of web applications. Without the knowledge of user it can allow an attacker that will perform unauthorized activities by which an attacker can manipulate your private data, Like if any user can send email to his colleague then the vulnerable web application do the same thing. Still it is hard to detect CSRF Vulnerability. In this paper we will understand the vulnerability of CSRF and how to detect CSRF Vulnerability in URL of web application using Python Script.

Index terms- CSRF, CSRF Detection Tool, Python, OWASP, Cyber Security

I.INTRODUCTION

Cross-site Request Forgery referred as CSRF/XSRF is the OWASP top 10 listed vulnerability found in most of web applications. CSRF is known as "SLEEPING GIANT" of the web-based applications vulnerability, just because of security missing in the web development. CSRF is also named as Confused Deputy Attack, Cross-site Reference Forgery, XSRF, Sea Surf, One Click Attack and session riding.

Without any Expertise tool, it can be a challenge to detect CSRF Vulnerability. It is considered as an active application layer attack. As with the help of new technologies, methods and tools the security of internet is increasing with the high speed, still the attackers find the vulnerability in the web applications and exploit them to carry out attack against servers and client.

The most dangerous attack today is accessing the confidential information that is linked with users account example: bank accounts, social media accounts, email accounts which has most sensitive information. Penetrating of these accounts may cause harm to the users in which they can lose their data and money. And this data is used by attackers to perform malicious operations on the cyber world. When user do any action on an internet like registration or filling up the form , the users data is sent on server by using GET or POST method of HTTP Protocol.

The browser sends the data/request according to the user's cookie. The cookies i.e. Session identification shows that how website store your privileges in your web browser and identifies who are you. the attackers taking advantage of this active session and send the malicious link to the user and when user click on that malicious link the CSRF attack happens on that particular users web browser. Thus the problem is that how to protect your active session that prevent you data or secret information from stealing.

There are many program of software that are testing the web applications that detects the security vulnerabilities. This software program is known as Security Scanners. There is branch of Information Technology i.e. Web Application Security that mainly deals with the security of the applications, Servers, Web services. As vulnerability exposes are growing daily, the developers also understand the vulnerability present in their software. There is huge problem in testing the software by hand, so there needs to be an automated vulnerability scanner that detects the web vulnerability by its own.

There are many different kinds of request forgery attacks:

- a. Man-In-The-Middle Attack: when an attacker hijacks the request and act as a server for forwarding request to actual server this can only be avoidable using SSL.
- b. JavaScript injection on the target site: If any attacker implements JavaScript on the page, they can perform every activity which the user can perform on their system.
- c. Browser-In-The-Middle Attack: The attacker target the browser of the user, when user left the machine after completing his/her work on the machine browser. This vulnerability can permit doing everything that the user can actually do in their browser. This is done when users logs out.
- d. Request forgery from another site: This is called actual CSRF attack. It is also called CSRF because it includes two sites to accomplish.

II. OVERVIEW OF CSRF ATTACK

CSRF attack is classified as pre-authentication and post-authentication CSRF Attack. As post-authentication CSRF is known from 2001, received lot of attention from web community and pre-authentication in not mentioned in OWASP Testing Guide although many exploits have been reported. The First CSRF attack was registered in 2001.

CSRF Attack is rarely known to developers and the consider it as XSS attack and some of them consider it as XSS Mitigations can work on this Attack. But it is different from XSS attack and it requires different methods and technique to detect this attack.

Lets take an example of CSRF attack, suppose an user is logging into an bank account on website using username and password, Server will validate the authority to the user and will provide a session to client. Attacker will send illegal request to attract them to a fake link which is on other third party untrusted server. Whenever use will click on that link the CSRF attack will get started.

CSRF Attack can also exploit the authentication mechanism of the targeted web applications. The problem with this is that web authentication assures request from site came from certain user’s browser, actually it does not know that the request came from is actual request or authorized the request.

If a server has CSRF vulnerability and accepts GET request, then the CSRF attacks are possible without use of JavaScript and if server is accepting POST request then the JavaScript is required to send automatic POST request from attacker’s application to target application.

CSRF Attack can simulate valid requests, Active XSS, SQL Injection and call web services in some cases of CSRF attack.

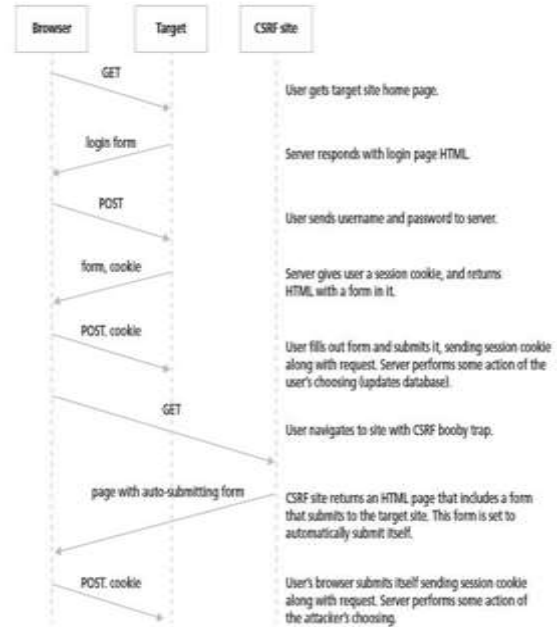


Fig. 1. What happens in CSRF Attack?

A. CSRF attack at URL

In URL password changing happens something like this. Example: Victims old password is rsu321. Now if we change new password to abc321, if URL contains following two factors: 1. Passwd-new = abc321 2. Passwd-config = abc321 and are separated by “&”. The attacker can operate the URL string by Address bar or using CURL. There are many other methods that can perform CSRF attack like VBScript/ActionScript/JavaScript/HTML etc.

B. CSRF Classification

CSRF is the vulnerability in which an attacker can perform unauthorized activities without the knowledge of the user. This vulnerability is classified in following:

- 1. Reflected CSRF: In reflected CSRF vulnerability the attacker use a system outside the application to expose the victim to exploit content.

2. Stored CSRF: Stored CSRF vulnerability is the vulnerability where an attacker use application itself for providing some content that directs the victims browser back to the application. These vulnerability are most likely to succeed.

C. CSRF Mitigation

There are number of methods, tools and techniques to prevent the CSRF attacks. Some of the best way to prevent the attacks are:

- 1 Secure your username and passwords.
- 2 Deny to remember the passwords in browser.
- 3 When applications not in use log off the application.

III. SYSTEM

Day-to-day the number of vulnerability exposes is increasing. The developers do not understand the security vulnerability of their applications. For secure coding and testing a dedicated person is required for this and it's not possible easily so, Here we need an automated system that can detect the vulnerability for the web applications which provide and speedy and accurate result.

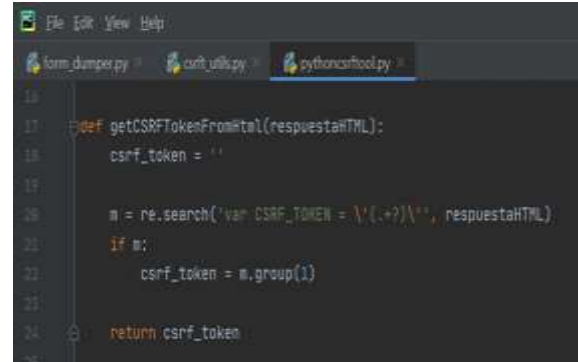
A. How today's software system finds the CSRF? CSRF is clear design flow vulnerability. Some Address Security Specifications are omitted like in password-reset CSRF Example, here" Only Authenticated entity can reset their own password" is missing specification. CSRF is mostly identified by DAST (Dynamic Analysis Software Testing). But we can still see the source code in which it is possible to find weak design-patterns like Password-reset example. But still the DAST or human source-code reviews can not identify the CSRF vulnerability. It seems like that DAST only scans the vulnerability in following manners: 1. Scanner that ignores CSRF Vulnerability. 2. Most users turn off the CSRF testing specification or they have over emotional affection with many more result they possibly can sort. 3. Try to replay all non-idempotent request and all successful replays as CSRF vulnerability. Here we are going to create one python system that will detect the CSRF vulnerability. So Let's have some basic introduction about python language.

B. Python

It is easy learning and powerful language used mostly in today's web applications. It is high-level data

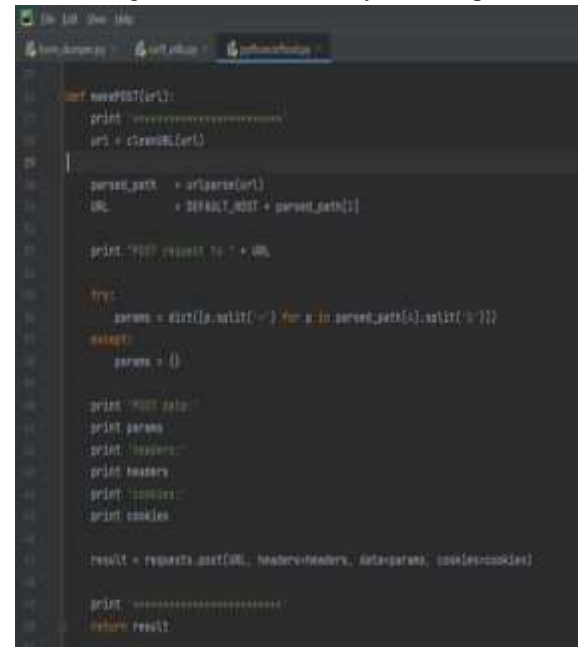
structure language. We can learn more about python from this link <https://www.python.org/>.

Here is the script that we are going to use for detecting CSRF Vulnerability. There are some parameters used in script for getting URL of the web application and scanning the vulnerability.



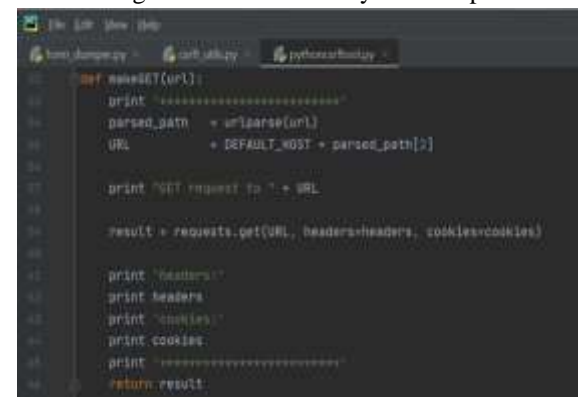
```
File Edit View Help
form_dumper.py  csrf_utils.py  pythoncsrftool.py
17 def getCSRFTokenFromhtml(respstaHTML):
18     csrf_token = ''
19
20     m = re.search('var CSRF_TOKEN = \'(.+)\'', respstaHTML)
21     if m:
22         csrf_token = m.group(1)
23
24     return csrf_token
```

Fig. 2. CSRF Token Python Script



```
File Edit View Help
form_dumper.py  csrf_utils.py  pythoncsrftool.py
17 def makePOST(url):
18     print "*****"
19     url = cleanURL(url)
20
21     parsed_path = urlparse(url)
22     URL = DEFAULT_HOST + parsed_path[0]
23
24     print "POST request to " + URL
25
26     try:
27         params = dict([a.split('=') for a in parsed_path[2].split('&')])
28     except:
29         params = {}
30
31     print "POST data:"
32     print params
33     print "headers:"
34     print headers
35     print "cookies:"
36     print cookies
37
38     result = requests.post(URL, headers=headers, data=params, cookies=cookies)
39
40     print "*****"
41     return result
```

Fig. 3. POST URL Python Script



```
File Edit View Help
form_dumper.py  csrf_utils.py  pythoncsrftool.py
17 def makeGET(url):
18     print "*****"
19     parsed_path = urlparse(url)
20     URL = DEFAULT_HOST + parsed_path[0]
21
22     print "GET request to " + URL
23
24     result = requests.get(URL, headers=headers, cookies=cookies)
25
26     print "headers:"
27     print headers
28     print "cookies:"
29     print cookies
30     print "*****"
31     return result
```

Fig.4. GET URL Python Script

IV. LITERATURE REVIEW

There are many system that provide protection against CSRF attack, these solutions have different methods. We have reviewed some literatures that detect the CSRF vulnerability.

In “Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery”, the author parimala have explained about the CSRF vulnerability and how to detect the CSRF vulnerability using Python Script. In their research they have made a tool that is web based and it detects the vulnerability in very efficient way.

In “CSRF tool: Automated Detection and Prevention of a Reflected Cross-Site Request Forgery”, author Omar Batarfi have mentioned about the CSRF detection and how they prevent the CSRF vulnerability using the CSRFtool. We have surveyed from these that we can detect and also can we prevent the CSRF vulnerability using Python Script.

In “CSRF Vulnerabilities and Defensive Techniques” by Rupali D. Kombade and Dr. B.B. Meshram they have mentioned the CSRF vulnerability and the techniques for preventing the CSRF attack. They have mentioned actually how the HTTP GET and POST method are using CSRF attack to be done. They have compared the all other techniques with their system.

In 2011 a paper titled “A Study of the Effectiveness of CSRF Guard” has discussed one of the strategies that has attempted to prevent and block CSRF attack. The protection that the CSRF Guard has offered against the CSRF attack deepened on the token generation and validation.

V. RESULT & DISCUSSION

By testing this vulnerability detection tool we have got the idea about how to test the vulnerability and where to secure the code at development side. With less number of verification, the detection tool detects the vulnerability. Thus also we can develop a tool or system that will further detect all other OWASP TOP 10 Vulnerabilities that will help developers to test their applications.

VII. CONCLUSION

This paper thus concludes that Cross-Site Request Forgery is vulnerability of OWASP top 10 and is found in most web applications thus to detect that vulnerability we can make a tool that can detect the vulnerability. Only our main aim was to detect the vulnerability using tool. Thus we can detect the CSRF attack using python script that can help us lot in many other vulnerability detection. CSRF is OWASP top 10 Vulnerability which access the data of users in an unauthorized way. So, this paper have worked upon the Vulnerability, how it works and how the flow goes and then after how to detect the vulnerability using python script.

VIII. ACKNOWLEDGMENT

I would like to thanks our Professor Dr. Ravi Sheth who guided me in my research work and also helped me a lot and I would also like to thanks sir Falgun Rathod who guided me at each and every step of my work.

REFERENCES

- [1] Omar A. Batarfi, Aisha M. Alshiky, Alaa A. Almarzuki, Nora A. Farraj, “CSRFtool: Automated Detection and Prevention of a Reflected Cross-Site Request Forgery”, 2014.
- [2] Aung Khat, “A Most-Neglected Fact about Cross Site Request Forgery”, by YGN Ethical Hacker Group, <http://yehg.net>, 2010.
- [3] Vandana Dwivedi,, Himanshu Yadav and Anurag Jain, “SQLAS: TOOL TO DETECT AND PREVENT ATTACKS IN PHP WEB APPLICATIONS”, 2015.
- [4] Avinash Sudhodanan, “Large-Scale Analysis & Detection of Authentication Cross-Site Request Forgeries”, 2017.
- [5] Parimala G, “Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery”, National Conference on Mathematical Techniques and its Applications (NCMTA 18) IOP Publishing IOP Conf. Series: Journal of Physics: Conf. Series 1000 (2018) 012125 doi :10.1088/1742-6596/1000/1/012125.
- [6] Rupali D. Kombade, Dr. B.B. Meshram, “CSRF Vulnerabilities and Defensive Techniques”, I. J. Computer Network and Information Security, 2012, 1, 31-37 Published Online February 2012

- in MECS (<http://www.mecs-press.org/>) DOI: 10.5815/ijcnis.2012.01.04.
- [7] William Zeller and Edward W. Felten, “Cross-Site Request Forgeries: Exploitation and Prevention”, 2008.
- [8] <https://brochure.getpython.info/>
- [9] Sentamilselvan K, “Survey on Cross Site Request Forgery (An Overview of CSRF) ”, IEEE - International Conference on Research and Development Prospects on Engineering and Technology (ICRDPET 2013) March 29, 30 - 2013 Vol.5 ISBN: 978-1-4673-4948-2 © 2013 IEEE.
- [10] Virginia Mary Nadar, Madhumita Chatterjee, Leena Jacob, “Detection Model for CSRF and Broken Authentication and Session Management Attack”, Virginia Mary Nadar et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (4) , 2016, 1801-1804.
- [11] Shaikh, Roshan, Defending Cross Site Reference Forgery (CSRF) Attacks on Contemporary Web Applications Using a Bayesian Predictive Model (March 1, 2013). Available at SSRN: <https://ssrn.com/abstract=2226954> or <http://dx.doi.org/10.2139/ssrn.2226954>
- [12] Jesse Burns, “Cross Site Request Forgery An introduction to a common web application weakness”, ©2007 Information Security Partners LLC.
- [13] Roshan Shaikh, “Defending Cross-Site Request Forgery (CSRF) Attacks on Web Applications”, 2019.
- [14] <https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/>
- [15] <http://buddie-hackersguide.blogspot.Com/2011/04/what-is-cross-site-request-forgery.html>
- [16] <https://www.whitehatsec.com/blog/whitehat-securitys-approach-to-detecting-cross-site-request-forgery-csrf/>
- [17] <https://portswigger.net/web-security/csrf>