Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery

Parimala G1  Sangeetha M2  R AndalPriyadharsini3  
Department of Information Technology, SRM Institute of Science and Technology

E-mail : parimala.g@ktr.srmuniv.ac.in

Abstract. Now day’s web applications are very high in the rate of usage due to their user friendly environment and getting any information via internet but these web applications are affected by lot of threats. CSRF attack is one of the serious threats to web applications which is based on the vulnerabilities present in the normal web request and response of HTTP protocol. It is hard to detect but hence still it is present in most of the existing web applications. In CSRF attack, without user knowledge the unwanted actions on a reliable websites are forced to happen. So it is placed in OWASP’s top 10 Web Application attacks list. My proposed work is to do a real time scan of CSRF vulnerability attack in given URL of the web applications as well as local host address for any organization using python language. Client side detection of CSRF is depended on Form count which is presented in that given web site.

1. Introduction

Web applications are stereotypically established with hard time limitations and are often positioned with security vulnerabilities. Web vulnerability scanners can aid to discover these vulnerabilities and are current tools amongst developers of web applications [1].Web applications vulnerabilities permit attackers to achieve malicious actions that range from ahead unauthorized account access to gaining sensitive data[1].Earlier, many standard websites hacked by hacker. They are now vigorous and always want to hack weak websites and leakage data. That’s why security testing is vital for web applications. Web application security scanners will take care of it [1].There is some software program for doing black box testing on Web Application to detect security vulnerabilities. That software program is called as Security Scanner [1][2].Web application security is a branch of “Information Security” that deals specifically with security of websites, web applications and web services [2].Every year, vulnerability exposes are tending to be growing. Scanning program developers frequently not understand the presence of vulnerabilities in that software [1][2].Obtaining a secure program software enthusiastic software program tester is required. They are testing the software by hand and search for the presence of vulnerabilities. But the worst part is performing software testing by hand. It could be huge troubles. So every developer needs an automated web vulnerability scanner tool for detect the software as its own. [1].

Session management web vulnerabilities [2] are considered as vulnerability that is repeatedly find on web applications. These vulnerabilities involve in session fixation, insufficient cookies attributes and CSRF. OWASP (Open Web Application Security Policy)Top 10, disputes on session management vulnerabilities are hierarchical on 2nd position, CSRF on 8th position[2]. Now browser side finding methods are established on cross origin policies that permit all third party white listed websites to perform requests to a trusted website. If the policies are
definite imperfectly then these tactics are not effective. These methods cannot emphasis on the recognition of CSRF attack where payloads locate in right-hand websites. There are several different kinds of request forgery attacks.

JavaScript injection on the target site - If attackers implement JavaScript on a page, they can do anything which user can do. This can be prevented by avoiding XSS vulnerabilities.

Man-In-The-Middle attack - when an attacker hijacks the request and act as the server to forward requests to the actual server. This is only avoidable using SSL.

Browser-In-The-Middle attack - when the attacker hijacks target’s browser, the victim has left the machine while actuality logged in to a site.

This permits doing anything the actual user can do. This can done when users is to be log out.

Request forgery from another site - This is what a Cross-Site-Request-Forgery (CSRF) attack is. It is called a Cross-Site-Request-Forgery because it involves two sites to accomplish. This happens when an attacker on another site sets something up so that a user submits a damaging request for the attacker. The action is achieved as long as the user has authorizations to do so on the target site. This kind of attack cannot be disallowed by the use of SSL.

Related Works

1.1 Overview of CSRF Attack

Cross-Site Request Forgery (CSRF) is also called as session riding and one click attack. When a malicious web site, web forum or email creates a target’s web browser to perform an unwanted performance on a reliable web page [3]. CSRF attack mostly involves in three things. They are a trusted site, a cruel site and targeted site. This target victim grips a lively session on a reliable website and concurrently the victim visits a malicious site. That malicious website adds a HTTP send request for the reliable site into that victim user session negotiating its integrity. For example one victim logs in a website A. After logging in to that website, the victim is staying on the legal session. If suppose the victim also desires to brows a website B without logging out from the previous website. But here the website B, attacker has dispatched a malicious link in that website B. This is capable to forward HTTP request to the website A to request to achieve some legal action but all needs a legal valid session. If the victim user presents on the website B and then by mistake gets into the malicious link then the HTTP request will direct to the website A where victim uses the legal sessions of the victim to achieve valid legal action on the website A. The main pretend of attacker is to post on the malicious link and invite target to click on that link. Then posting the malicious link on the website that contains the image tags [5].

Here is the example to cruel link which is using image tag. No one browser does check image tag for any particular type of image extension. It can point to any webpage. We have perceived the example of the malicious link which is in above. So when going for demonstration, Consider as the victim user named Priya, she is logging her bank website for e-banking. This bank has a web page for transferring money to another user's bank account. The form of this transferring action looks like this.

```html
<form action="http://www.mybank123.com/transfer.php method="GET">
  <input type="text" name="amount" for amount
```


If someone needs to transfer money, they should submit the form. After the submitting the form, the next page will be like transferring. php and URL will like this:

http://mybank123.com/transferring.php?acct=sender&amt=amount&for=receiver

This URL will come after submitting the form. If you are capable to produce the same action by posting this URL on the address bar, it means the website is vulnerable to attack.

For violent, website is considered as a genuine user, the hacker should be known about the structure of the website forms. If attacker knows how website use form for this action, attacker can do a link that can directed this valid HTTP request to the website. Then attacker will generate a link like this:

<img src="http://mybank123.com/transferring.php?acct=sender&amt=amount&for=recipient">

In case the user in logging stage and they clicks on the malicious link then that form will be submitted automatically, the action will achieve [6].

And several methods to forward the link to that target site.
1. Posting image link by a third party website.
2. By Email.
3. On rapid messaging.

The furthermost common behavior to implement CSRF attacks are presented by using a JavaScript image object and HTML image tag. Normally an attacker can implant these links into websites or an email when the victim adds the malicious page or an email, then they achieve HTTP request to URL link. Below is a list of the collective methods that the attackers can try forwarding the request.

![Graph showing percentage of different types of web applications being attacked by CSRF reported from NVD](Image)

**Fig 1**

**Fig 1:** Percentage of different types of web application being attacked by CSRF reported from NVD

### 1.2 CSRF Attack at URL

Now in URL the password changing happening can see that URL itself.

Egg: victim’s old password is – priya321

Now if this goes to change by replacing the new password like test321

If the URL has contain these below two factors and separated by a “&”

1) passwd-new=test321
2) passwd-conf=test321
This is the example of worst enactment how to change the password on web applications for the subsequent below reasons:
Here HTTP is being handled in its area of HTTPS that means this password change was in vibrant test.
Attacker could operate the URL string using the address bar or curl to change the password. Some other methods are in HTML /VBScript/ActionScript/JavaScript and some other markup languages are also make the users browser perform isolated requests to perform CSRF attack

2. Existing System

Every year, number of vulnerability exposes is tending to be growing. Here Software program developers do not understand the existence of security vulnerability exposures in their software. To obtain secure software, a dedicated program testing person is desired. They are manually exam the websites and search for the presence of vulnerabilities. Unfortunately, manual software testing might be a big pain [1]. We need an automated vulnerability scanner tool for testing the software constantly.

Web applications are integral part of our life nowadays. At the same time, the attacks using web application vulnerabilities and the damage caused by them are increasing. However, it is difficult to develop completely secure Web applications. Furthermore, checking all web application vulnerabilities by hand is difficult. These are time consuming, error-prone and costly. Therefore we consider that a good automated tool to detect Web application vulnerabilities is needed. And very few tools are available for detecting CSRF vulnerability attack on browser side.

We are looking into some of the research carried out in exiting vulnerability scanning tools.

In most of the vulnerability scanning tools are used to scan the malicious things and generate the report for that. Some available tools in market, they can be acquired by paying money.

2.1 Problem Statement

There are many tools are available for detecting CSRF Sleeping Giant. Unfortunately no one tool is there for detecting this without false positive. Still the top most companies are doing manual methodology for finding CSRF.

3. Proposed System

My proposed CSRF vulnerability detection tool will be detect CSRF in very effective manner without any false positive. My scanning tool and user interface are designed by using python language.

3.1. Analysis of CsrF Using Dvwa

I have used My Own Server (WAMP) and My Own Website (DVWA).After doing every Analysis; I have found something from this DVWA website about CSRF Attack,

First at low level security, Proof of Concept (POC) has been created by getting Form ACTION and creates new Form with targeted ACTION link on notepad. CSRF attack can be done on the same browser. After creating POC on notepad and compiling it in same browser, CSRF has been done successfully this method only applicable for LOW level security from CSRF. In this low level security, there is no CSRF Token.

Then Medium level security, The URL has the following factors separated by using "&".

1] Passwd-new=abc321
2) Passwd-conf=abc321

These DVWA’s examples are considered as a bad enactment how to make change a password on a web application for the next reasons:

1) HTTP is being handled its place of HTTPS, which means this password change was in unencrypted text format.

2) An invader could deploy the URL string using the address bar or curl to change the password (XSS attack – Session ID), and by changing MD5 value which is in Php Admin database. So by using curl to change the password remotely deprived of a browser. Here also there is no CSRF security tokens.

Finally, High level security on DVWA website has Anti CSRF token. In GET parameter, it checks anti-CSRF–token. So bypassing is very difficult in this level. So my proposed work is fully depends on Token analysis.

Fig 2

3.2. Project Work

“Vulnerabilities Scanner “basically software designed in Python that aims to scan the Website’s and finding out vulnerabilities that sustains in the websites and providing report to the client.

Scanner is going to take the URL of the website as and input, to test for vulnerabilities as per the selected vulnerability

For a valid URL, it will crawl the Website with the help of our own developed Crawler basically made in Python. The Crawler will parse the website. The output of this phase is multiple frameworks,
form details and platform used in developing web pages of respective website. After parsing the website, Scanner will test all the forms and links of the website for the vulnerabilities.

In my project work, I have used list of parameters to detect CSRF vulnerabilities. The parameters are token, CSRFTOKEN, authenticity_token, csrfmiddlewaretoken.

Here I have used,

1) beautifulsoup- This is Python library for pulling data out of HTML and XML files
2) httplib- The client side of the HTTPS and HTTP protocols are described here. It is typically not used directly; here we use urllib to lever URLs that uses HTTP and HTTPS
3) urllib- It delivers a high-level interface for enticing data across the WWW. In particular, it takes Universal Resource Locators instead of filenames, the function urlopen() is alike to the in-built function open(). Some limitations apply it can only open URLs for reading.

```python
import urllib  # URL Library
import re  # Regular Expression Library
from bs4 import BeautifulSoup  # Beautiful Soup Library
import http

After importing libraries, my work has been followed by below steps,

1) Check user input (URL/Local host file) : First form value should be counted from given input. That may be a URL or Local host file.

2) Counting Number of Forms: If the given input has any form count then only CSRF vulnerability checking will start. If the form count is zero then that is no possible to CSRF attack.

```python
mainpg=urllib.urlopen(url).read()
soupmainer=BeautifulSoup(mainpg,"html.parser")
count = mainpg.count ("<form")
if count == 0:
    print ("There are no forms found")
exit()

Print("""<formaction={}onsubmit={}>"""".format (count1,counter1))
```

Depending on the form count on the input site, we can do the analysis whether there is the chance to CSRF attack or not.

3) Token Analysis: Every form which is present in that website will go for TOKEN analysis test. And based on the TOKEN analysis test, my project work has been done. Token is nothing but a random value it has been generated from server because your session is still active in browser and browser has your session id. This is the reason the most popular CSRF protection is having another server supplied unique token generated and appended in the request. This unique token is not something which is known to browser like session id.

4) Checking the Random value for each Request: If Token value is not there in any website then we can predict that website is non-free from CSRF attack. In some cases, token value is available for that URL page; we can't say that site is fully free from CSRF attack. In that case, that token value of
current time request shouldn’t be matched with previous request token value. If the token value is to be same then these is the chance to happened Token stealing attack. It may cause CSRF attack. For that my project has checked whether Token value is changed for each request or not. If token value is changed for every request then it will be concluded as no CSRF attack vulnerability otherwise it will be concluded as CSRF vulnerability.

5) Same Origin Policy: Under the policy, if two web pages have the similar origin in a web browser it permits scripts delimited from one page to another page. An origin contains port number, URI scheme, and hostname. This policy thwarts a malicious script from an different websites through Document Object Model. To find out whether SOP is affected or not in that particular website. Access Control Allow Origin is the method to find out SOP. That can be done by using Http Request and Http Response [8].

Here the Structure of the TOKEN has been analyzed. By taking the array of token value will be passed if the source code is shown then that page is conclude as vulnerable.

If any action happened on URL then POST methods (UPDATE, DELETE, Etc...) will be changed so it will be allowed Direct Access Allow to do on that web site. If Direct Access Allow doesn’t check logout condition then that website will be concluded as vulnerable to CSRF.

These methods tolerate a specific consequence for recent web applications that widely depend on the HTTP cookies to keep authentic victim sessions. On a client site to avoid data lose, these data are maintained in a confidential and integrity way.

4. Implementation and Evaluation

User should select the check box URL or Local host file as input on first page. If user selects URL then next page will ask URL link for CSRF detection. If the user selects Local host file as input then next page will ask Local host link for CSRF detection.
After giving the link, user have to submit that.

Then it will show form number present in that given link which is given by user as input.

then user should give the number of form to scan for CSRF vulnerability and then submit it.
Fig 6: GUI final result after completing scanning process.

After submitting form number as input, then final result will be showed depends on Token analysis

5. Conclusion

Conclusion of this paper, CSRF is an active browser based web application weakness which is not familiar to the website developers, so the attacks may is happen in many websites. There are many current websites are facing this kind of vulnerable now-a-days. The only way to protect web application from CSRF attack is to use Random Token system. This is castoff by many web sites to a guard against CSRF For every request and session should have different token value which means random token value should be changed per every request and session. Otherwise it will be considered as CSRF vulnerability. And this attack will be active on same browser. So my tool is checking properly under these scenarios to detect CSRF. Depends on from count which is presented in website, my tool will start to check. If the form count is zero then there is no possible for CSRF otherwise scanner will start to check that page whether there is any token available or not. Finally my tool is working properly to check the CSRF vulnerability and showing the result.

6. Future Work

There are large number of tools to detect web vulnerabilities, however the range of web vulnerabilities are far more that there is no one tool to detect all the vulnerabilities. My future work is going to design a tool which can detect most of the top 10 OWASP vulnerabilities under one roof with high accuracy and speed.

REFERENCES

[1] “Evaluation of Web Vulnerability Scanners”The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications24-26September2015, Warsaw, Poland

[2] “A Vulnerability Scanning Tool for Session Management Vulnerabilities”978-1-4799-7996-7/14/$31.00 ©2014 IEEE

[3] “A Study of the Effectiveness of CSRF Guard” 2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing
[4] “CSRF: Attack and Defense “ McAfee® Foundstone ® Professional Services White Paper

[5] “Improved CSRFGuard for CSRF Attacks Defense on Java EE Platform “ The 9th International Conference on Computer Science & Education (ICCSE 2014) August 22-24, 2014. Vancouver, Canada.

[6] “Cross Site Request Forgery: A common web application weakness” 978-1-61284-486-2/11 $26.00 ©2011 IEEE

[7] “Client-Side Detection of Cross-Site Request Forgery Attacks “ 2010 IEEE 21st International Symposium on Software Reliability Engineering 1071-9458/10 $26.00 © 2010 IEEE DOI 10.1109/ISSRE.2010.12

[8] “Lightweight Server Support for Browser-Based CSRF Protection” IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 6, JUNE 2015

[9] “CSRF Vulnerabilities and Defensive Techniques” I. J. Computer Network and Information Security, 2012, 1, 31-37 Published Online February 2012 in MECS (http://www.meecs-press.org/) DOI: 10.5815/ijcnis.2012.01.04

[10] “Research and Design on Web Application Vulnerability Scanning Service” 978-1-4799-3279-5 /14/$31.00 ©2014 IEEE

[11] “An Analysis of XSS, CSRF and SQL Injection In Colombian Software And Web Site Development” 1 st Int’l Conf. on Recent Advances in Information Technology | RAIT-2012 |

[12] “Using Web Security Scanners to Detect Vulnerabilities in Web Services “ marco Vieira, Nuno Antunes, and Henrique Madeira CISUC, Department of Informatics Engineering University of Coimbra – Portugal

[13] “GUARDING WEB APPLICATION WITH MULTI - ANGLED ATTACK DETECTION” 2014 International Conference on Soft-Computing and Network Security (ICSNS -2015), Feb. 25 – 27, 2015, Coimbatore, INDIA