



# Cross-Site Request Forgery

## Pentesting Guide



## Table of Contents

Abstract.....	3
<b>What are cookies and Session ID?</b> .....	4
<b>What is an SOP?</b> .....	5
<b>Introduction to CSRF Attacks</b> .....	6
<b>Impact of CSRF vulnerability</b> .....	7
<b>CSRF Exploitation</b> .....	7
<b>Manipulating user-account details</b> .....	8
<b>CSRF over Change Password form</b> .....	12
<b>CSRF over Bank Transfer</b> .....	14
<b>CSRF Drawbacks</b> .....	17
<b>Mitigation Steps</b> .....	18
Conclusion .....	18
References .....	18



## Abstract

*You always change your account's password when you desire for, but what, if your password is changed whenever the attacker wants, and that if when you are not aware with it?*

In this report, we'll learn the basic concepts about Cross-Site Request Forgery (CSRF) attacks or how **an attacker forces the user** to execute some unwanted actions that they (users) never intended to.

**Disclaimer:** This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.



## What are cookies and Session ID?

Before jumping into CSRF attacks and how such attacks are executed, we need to understand some important terminologies that somewhere or the other empowers the applications to be secure.

### Cookies

Cookies are basically some small text files with a maximum of size 4KB that are **stored over on the client's browser** in the form of name-value pair. Cookies are majorly **used to track or monitor the client's activity** on the web application and **even stores** some sensitive data such as **username, session ID's, password preferences**, etc, which thus can be sent back to the server for an authentication request.

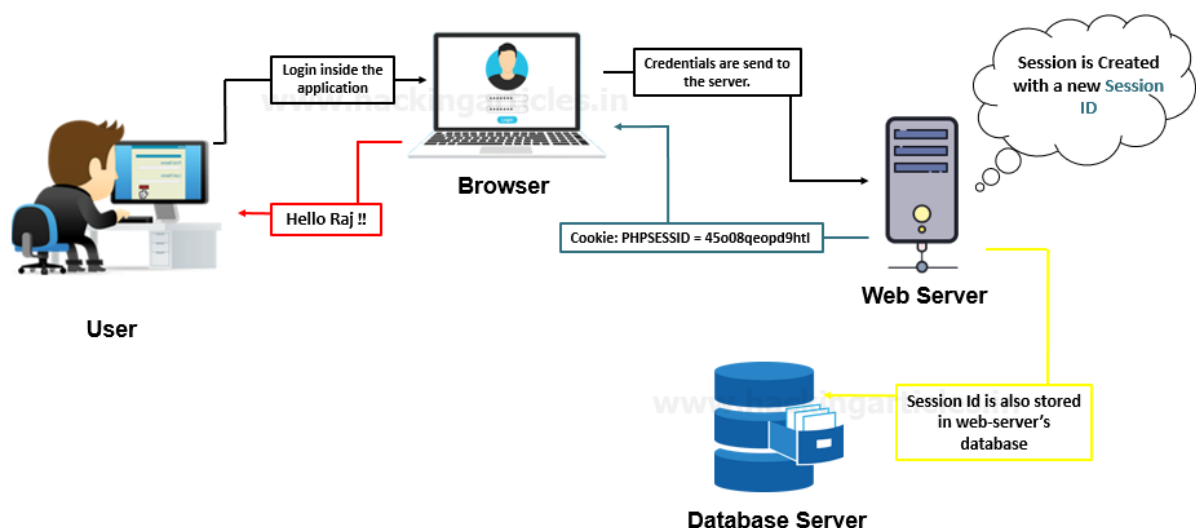
### Session ID's

Similar to the cookies, sessions are some small files too, but they are generated and stored at the server's end. Each session is associated with a session ID. Whenever **a user logs in**, thus **a session is created with a session ID**, thereby this session ID blends up into the cookie and stored at the client's browser and thus passed back to the web server whenever the browser makes an HTTP request.

So, whenever the client logs out or the browser is closed, these sessions are terminated. And with every new login, a new session with a session ID is generated.

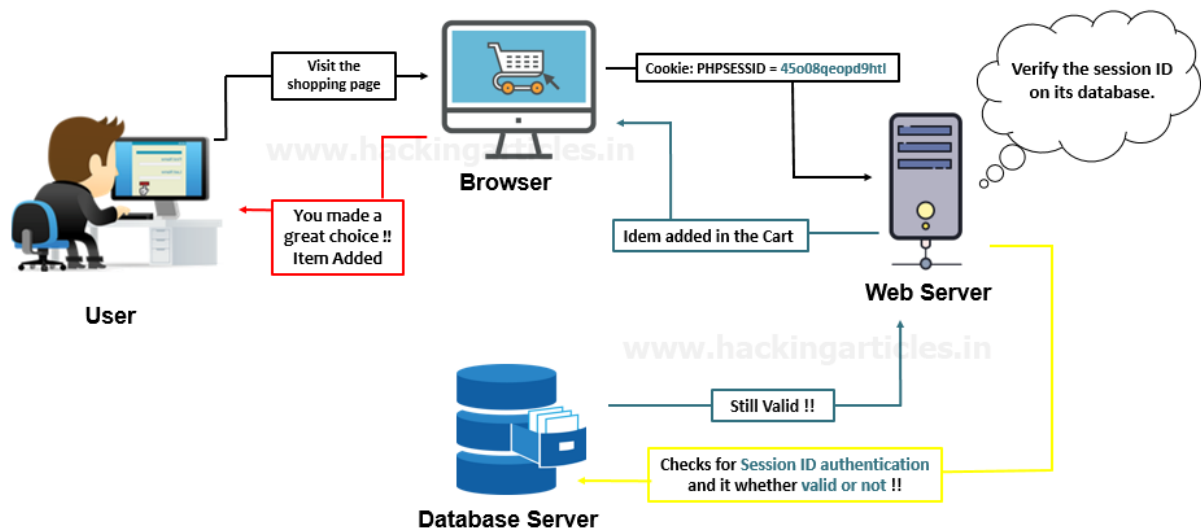
Let's make this clearer with the following case scenario –

*When a new user creates up his account over the web-application, the following procedure happens*





Now, when the user visits some other section of the same application, the session ID stored over in the user's browser is sent to the server for validation.



## What is an SOP?

**SOP** is an abbreviation for **Same-Origin Policy** which is one of the most important concepts in the web application security model. Under this policy, a web browser permits **scripts contained in a first web page to access data in a second web page**, but this occurs only when both the web pages are running over on the **same port, protocol and origin**.

**For example:**

Say the web-page "<https://www.ignitetechnologies.com/ceh/module1.pdf>" can directly access the content at "<https://www.ignitetechnologies.com/network/RDP/module7.docx>".

But it cannot access the data from "<https://www.ignitetechnologies.com:8080/bug/xss.pdf>". As the port has been changed between the two now.



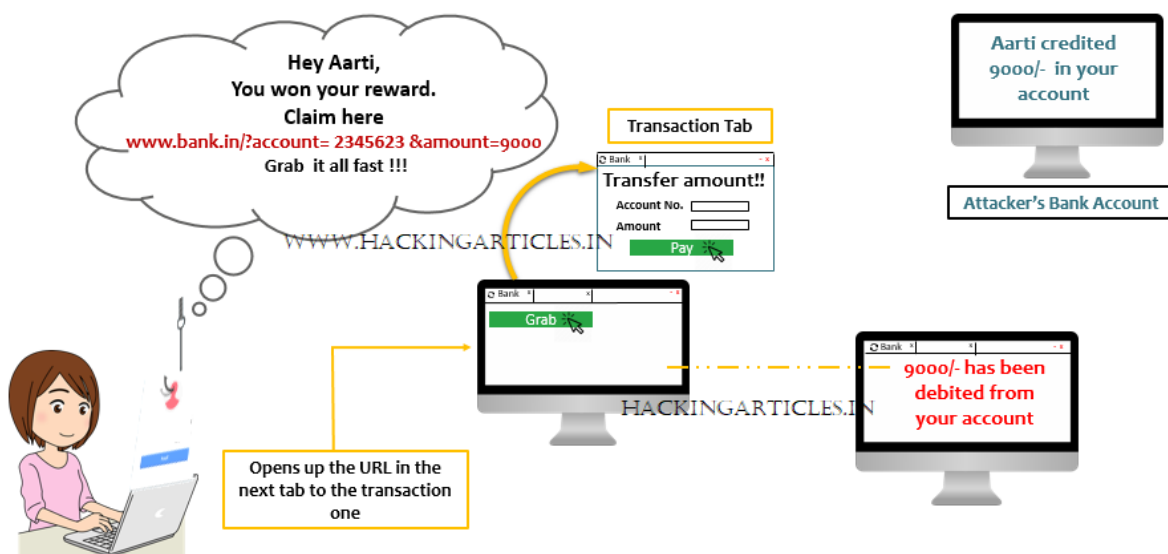
## Introduction to CSRF Attacks

**CSRF** is an abbreviation for **Cross-Site Request Forgery**, also known as **Client-Site Request Forgery** and even somewhere you'll hear it as a **one-click attack** or **session riding** or **Hostile Linking** or even **XSRF**, basically over in this attack, the attacker **forces a user to execute unwanted actions** on a web application in which they're currently authenticated.

To be clearer, let's check out the following case scenario:

Here the user **"Aarti"** receives a mail from the attacker, while she was doing a bank transfer to Mr Raj Chandel's account. She left the transaction incomplete and checks her mail over in the new tab.

There she got the generic URL, with a content stating that "You've received an amount as a reward". Now, as she opens the URL in the next tab, she got the **Grab button there**. The transaction page was over in the other tab and this malicious page is on the next one. As soon as, she hits the submit button, the query executes up and the amount has been deducted from Aarti's account without her knowledge, or concern.





## Impact of CSRF vulnerability

CSRF is an attack that forces the victim or the user to execute a malicious request on the server on behalf of the attacker. Although CSRF attacks are not meant to steal any sensitive data as the attacker wouldn't receive any response as whatever the victim does but this vulnerability is defined as **it causes a state change on the server**, such as:

- Changing the victim's email address or password.
- Purchasing anything.
- Making a bank transaction.
- Explicitly logging out the user from his account.


Therefore, this vulnerability was listed as one of the **OWASP Top10 in 2013**. And thereby now has been reported with a **CVSS Score of "6.8"** with **"Medium" severity under**

- **CWE-352:** Cross-Site Request Forgery (CSRF)

## CSRF Exploitation

Let's Start!!

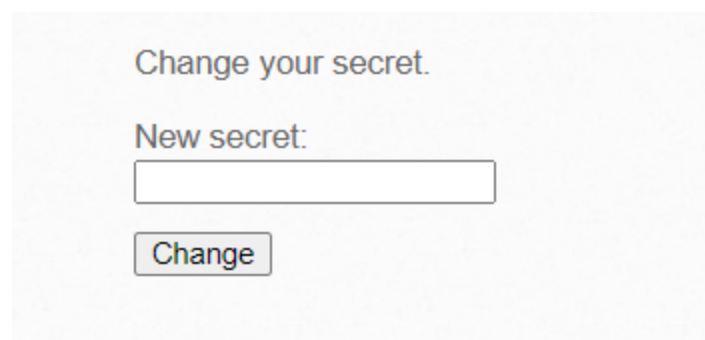
For this section, I've used bWAPP the vulnerable web-application and created an account with the user **Raj: ignite** to login inside the webserver.



The image shows a 'New User' registration form on the website www.hackingarticles.in. The form includes fields for Login, E-mail, Password, Re-type password, and Secret. The 'Login' field contains 'Raj', 'E-mail' contains 'Raj@ignite.com', 'Password' contains 'ignite', 'Re-type password' contains '\*\*\*\*\*', and 'Secret' contains 'who are you'. There is an 'E-mail activation' checkbox and a 'Create' button.

## Manipulating user-account details

Over in the above image, you can see that for creating a new user, the user “**Raj**” has set his **secret value** as “**who are you**”, now if he wants to change this secret value, he can change it from here.



The image shows a 'Change your secret' form. It has a label 'New secret:' followed by an empty text input field and a 'Change' button.

*But what, if his secret value gets changed with the attacker’s desired value without his knowledge?*

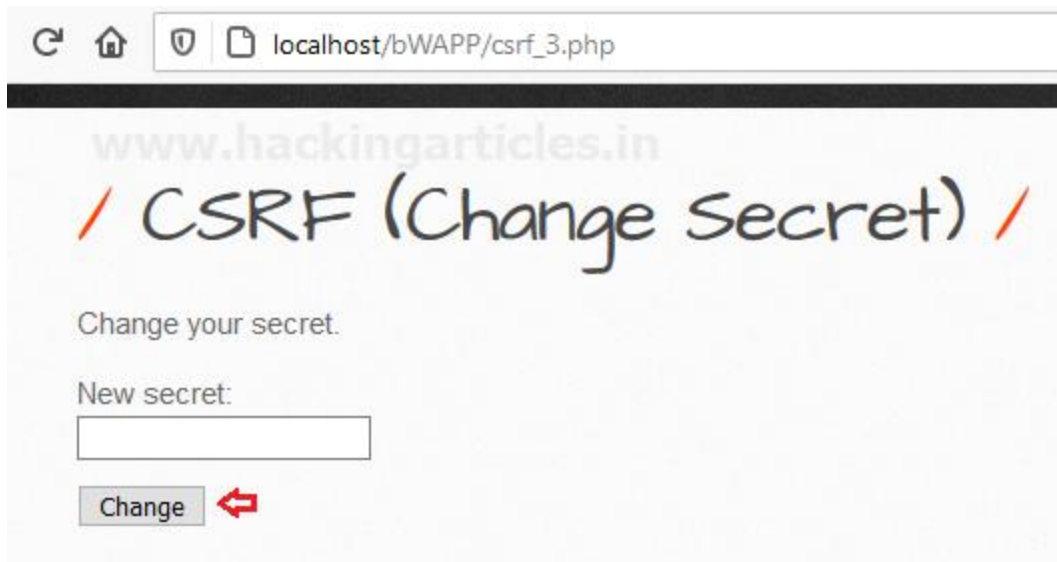
Let’s check it out how!!





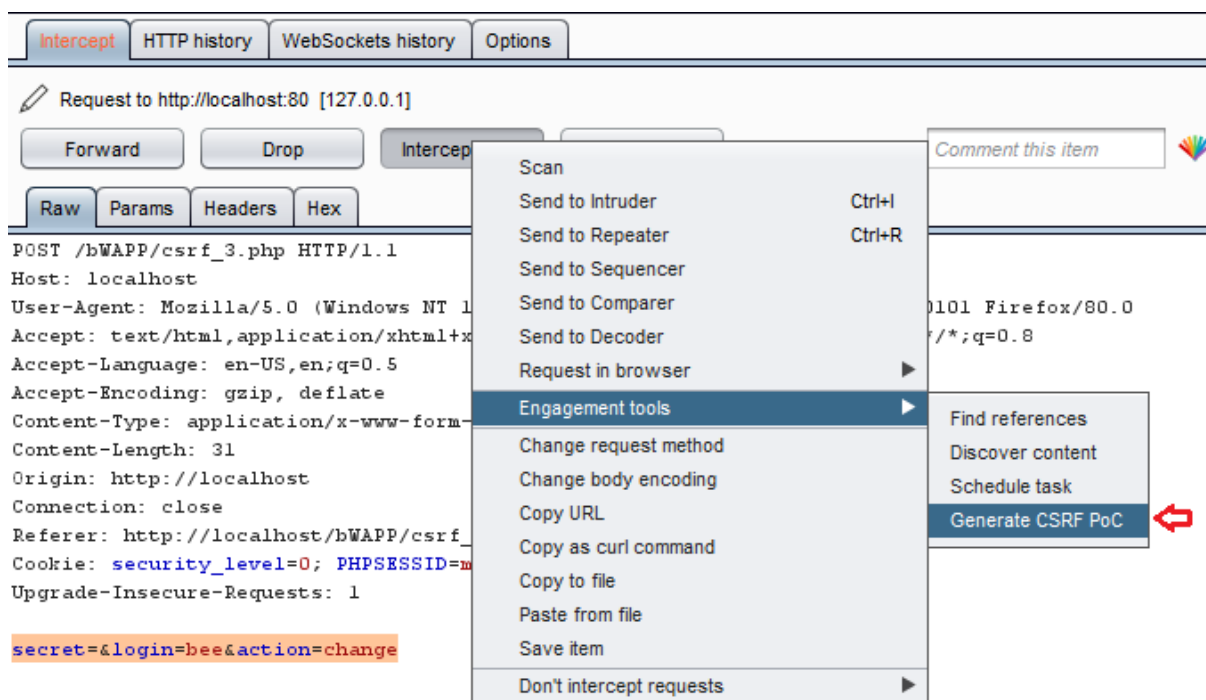
Let's open the target IP in our browser, and we will set the **"Choose Your Bug"** option to **Cross-Site-Request-Forgery (Change Secret)**.

Here, we'll be redirected to the web-page which is suffering from the CSRF vulnerability, where you can see that we're having the option to change the secret value.



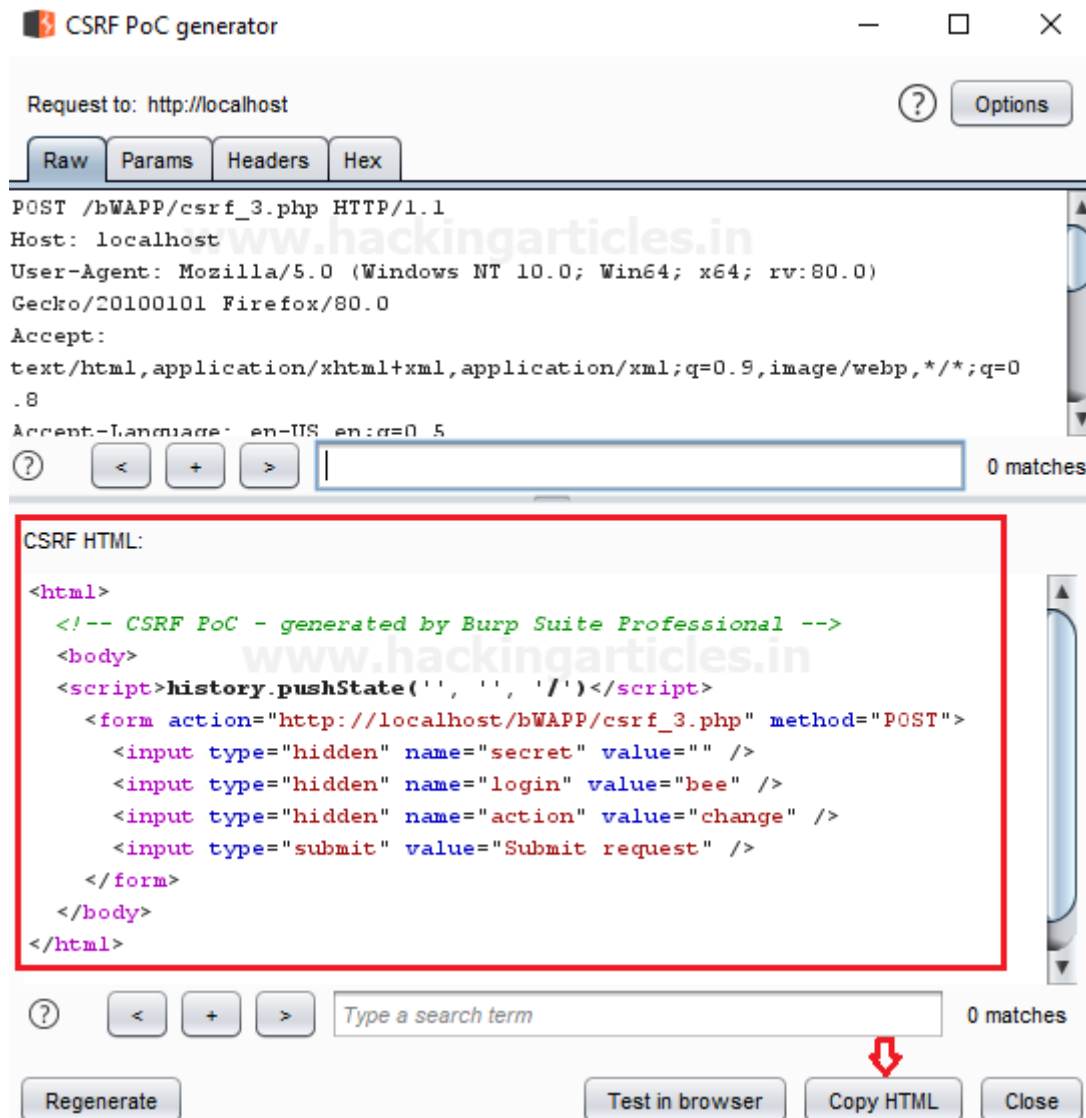
Now, let's **"hit the change button"** and **capture** the passing HTTP Request.

Form the below image, you can see we have successfully captured the request. Time to **create a Fake HTML form**, to accomplish this, do a right-click anywhere on the screen and select the **engagement tools** further hit the **Generate CSRF PoC** option.





This CSRF PoC generator will automatically generate an HTML form page. Now, click on **Copy HTML** in order to copy the entire HTML code, further past the copied data into a text file.



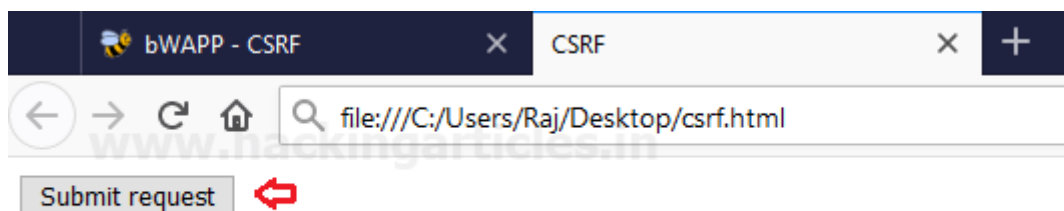
Great!! Let's manipulate the **secret value=""** parameter with **"hackingarticles"** and then we'll save this file as **"csrf.html"**. Moreover, we need to set the user name **"Raj"** as for whom the secret value will get changed.



```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://192.168.0.10/bWAPP/csrf_3.php" method="POST">
      <input type="hidden" name="secret" value="hackingarticles" />
      <input type="hidden" name="login" value="Raj" />
      <input type="hidden" name="action" value="change" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Now, we'll use the social engineering technique to share this csrf.html file to the targeted user.

As soon as the **victim Raj** opens up this **csrf.html**, there he will get a submit button, as he **clicks** over it, his secret will get changed without his (victim) knowledge.



From the below image, you can see that he got the message as **“The secret has been changed!”**, over with this similar way a successful CSRF attack can even change up the email addresses, usernames and the other personal information of the user.





## CSRF over Change Password form

“Change Your Password” feature is almost offered by every web-application, but many times the applications fail to provide the security measurements over such sections. So let’s try to exploit this “Change Password” feature over with the **CSRF vulnerability**, which is *one of the most impactful CRSF attacks where the user’s password will get changed without his knowledge.*

Back into the “Choose Your Bug” option select the **Cross-Site-Request-Forgery (Change Password)** and hit **hack button**.

Change your password.

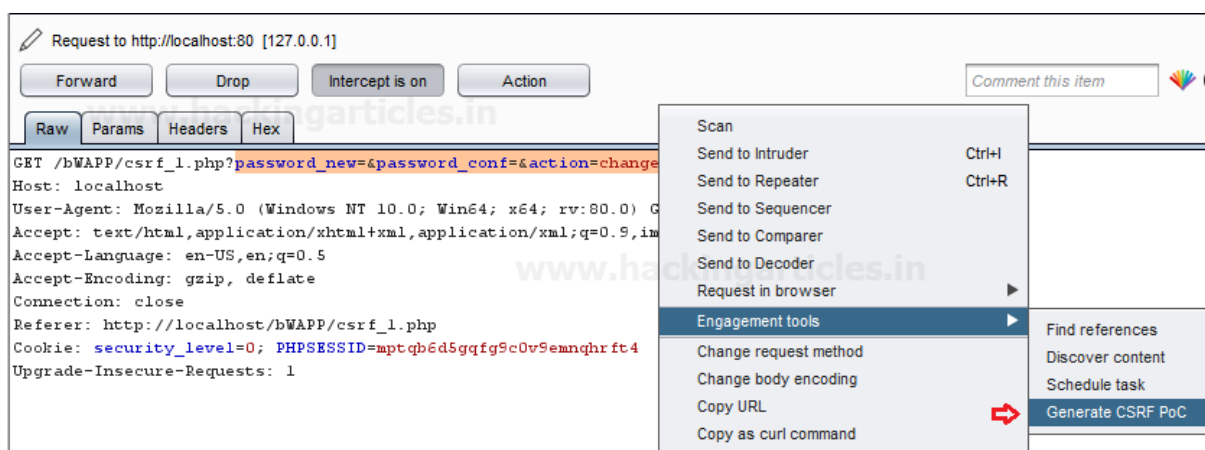
New password:

Re-type new password:

Change

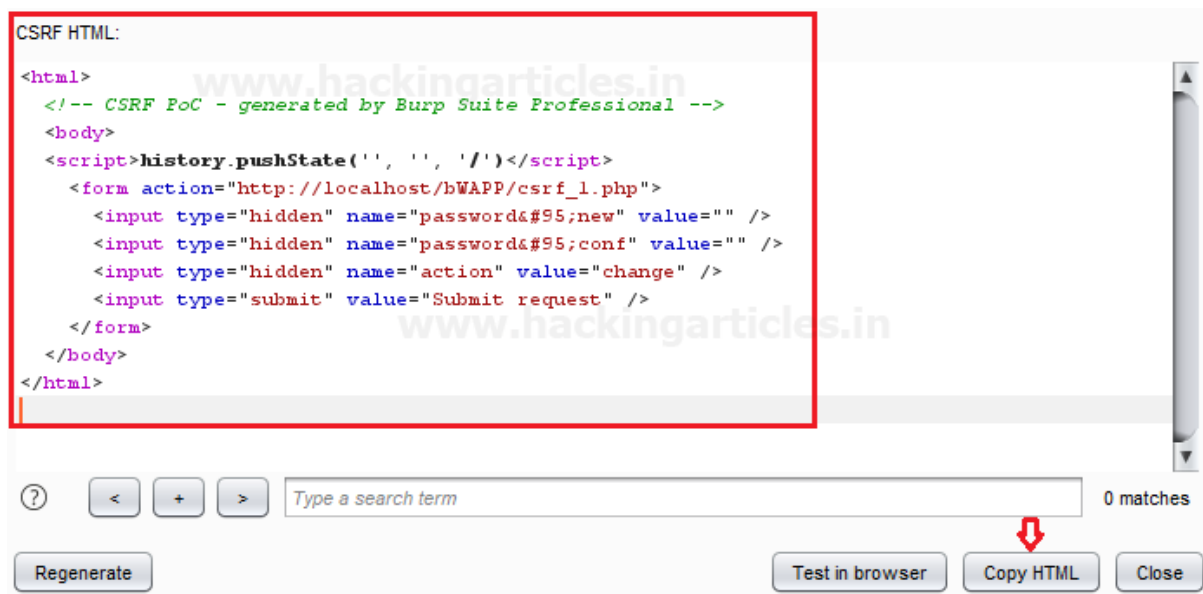
Now, let’s again “fire up the Change button” and **capture** the passing HTTP Request over in the **Burpsuite**.

Form the below image, you can see that we have successfully captured the request. Let’s follow up the same procedure to generate a “**forged HTML form**”.

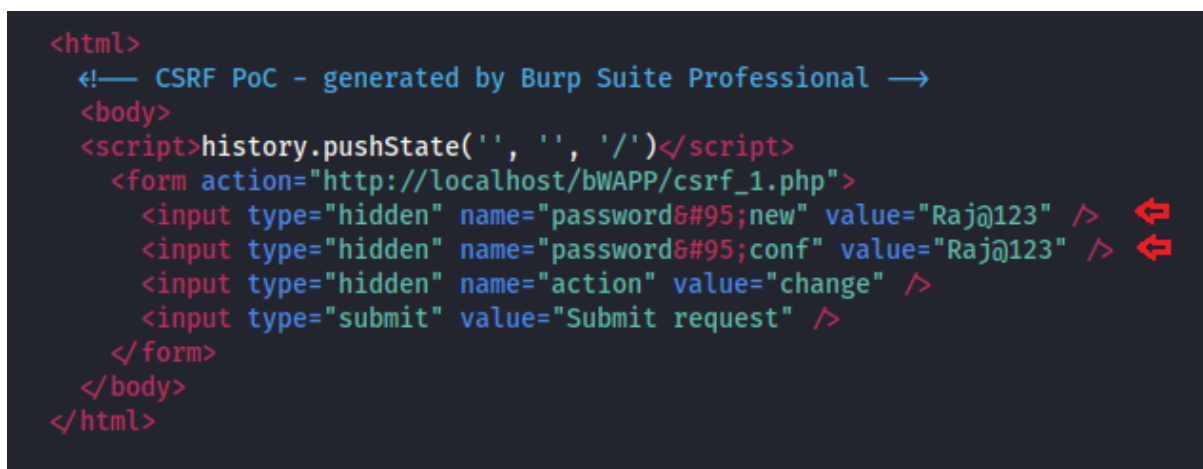




Now, click on **Copy HTML** in order to copy the entire HTML code, further past the copied data into a new text file.

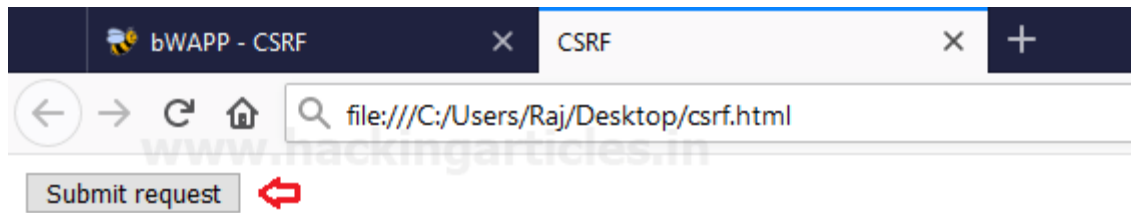


Once we've pasted the HTML code, let's now add the new password value (attacker's password) and the confirm password value, and then save the text document as csrf.html



Now, we'll again use the social engineering technique to share this csrf.html file to the targeted user.

As soon as the **victim opens** up this **csrf.html**, there he will get a **confirm** button, as he **clicks** over it, the password will get changed without his knowledge.



From the below image, you can see that the CSRF attack changes the old password set by user “Raj”.

Cool!! Now when he(victim) tries to login with the old password, he’ll get the error message.



## CSRF over Bank Transfer

*“You might have heard some scenarios where the money is deducted from the victim’s bank account without his/her knowledge, or a fraud has been taken place where the victim receives an email and as soon as he opens it up his bank account gets empty.”*

Wonder how CSRF is related to all this? Let’s check out the following attack scenario.

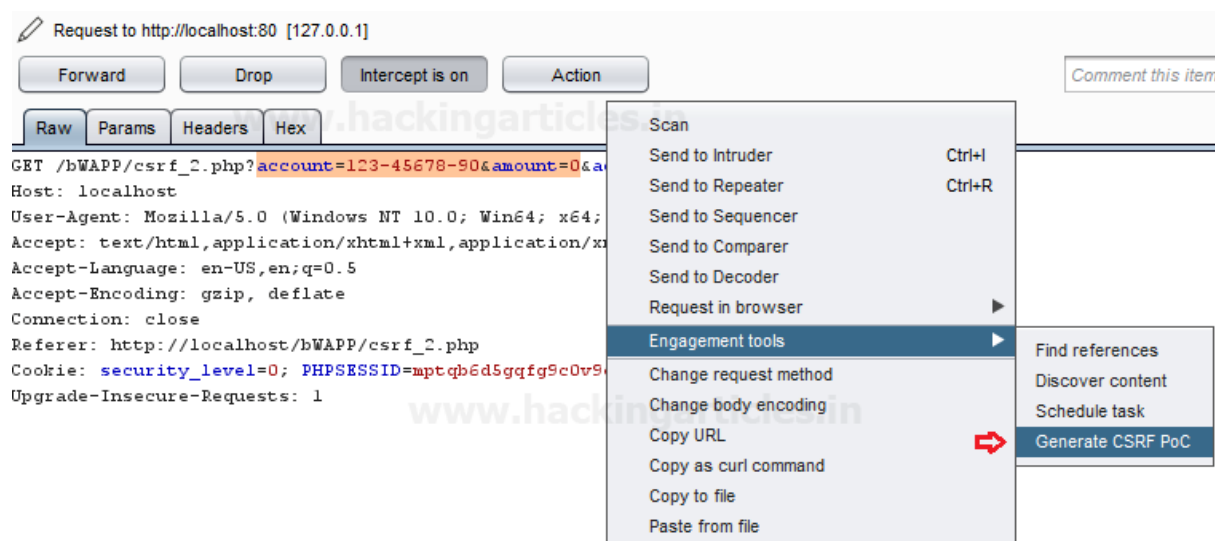


Login inside **bWAPP** again, then choose the next vulnerability **Cross-Site Request Forgery (Transfer Amount)** and click on the hack button.

Over in the below screenshot, you can see that the user “Raj” is having only **1000 EUR** in his account.

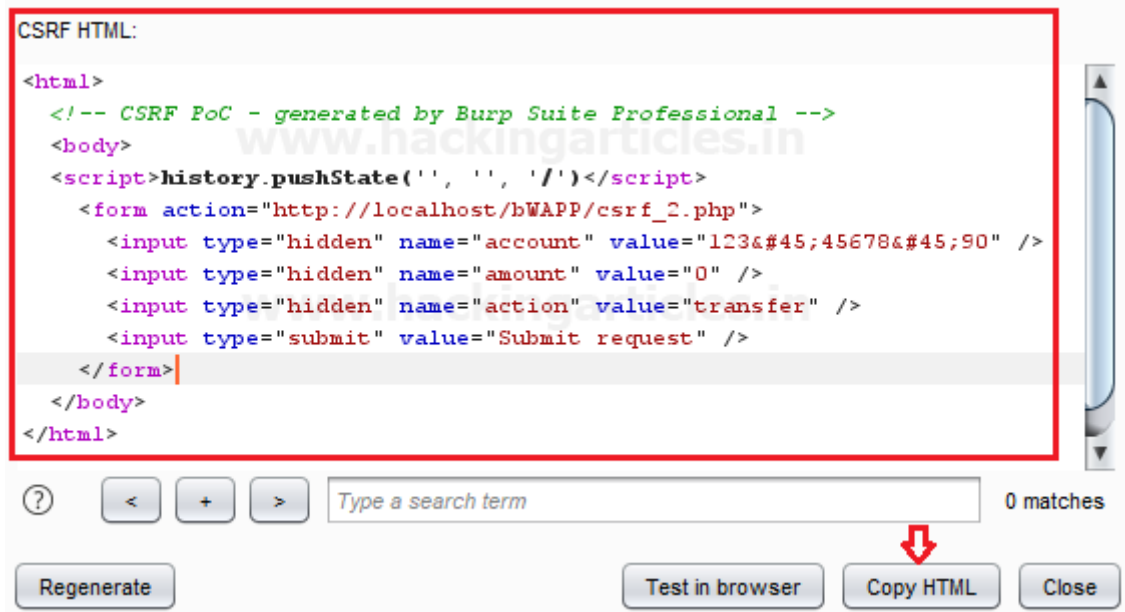
Let’s try to transfer some amount from this, as the account number is already there in the field.

The procedure for the **CSRF** attack is similar as above, use burp suite to capture the sent request of the browser and then share the request over on the **Engagement tools** section.

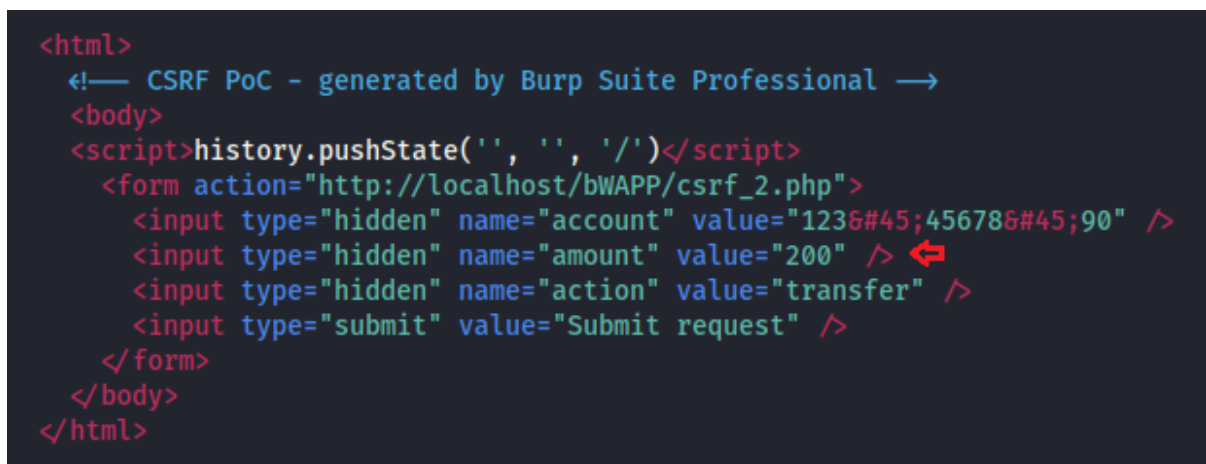


Again, it will create an HTML form automatically for intercepted data. Simply click on **Copy HTML** and paste it into a text file.

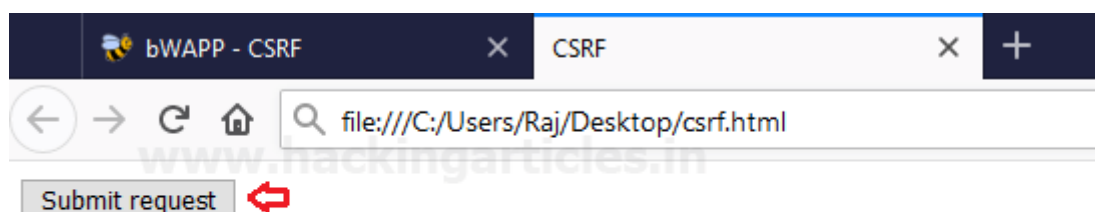




Great!! Time to manipulate the “**value**” field, add amount “**200**” (attacker’s desired amount) which to be transferred, save the text document as **csrf.html** and then share it with the targeted user.



As soon as the victim open’s this file up, and hit the submit button, the amount (entered by the attacker) will be transferred, without his (victim) knowledge.







From the given screenshot, you can see that the amount is left **800 EUR** in user's account which means 200 EUR has been deducted from his account.

CSRF (Transfer Amount)

Amount on your account: **800 EUR**

Account to transfer:  
123-45678-90

Amount to transfer:  
0

Transfer

Thereby over with such basic techniques, the attacker can make major changes over on the victim's account.

## CSRF Drawbacks

1. The CSRF attack is having a major drawback, as these attacks are only **successful when the attacker knows which parameter and what values are blended in the HTML form.**
2. And even these attacks need some good social engineering technique as sometimes the HTML forms require earlier used values as **“Current Password”** to change up the new password one.

Change your password.

Current password:

New password:

Re-type new password:

Change



## Mitigation Steps

- The developers should implement the use of “**Anti-CSRF tokens**”
- Use of **Same-Site cookies attributes** for session cookies, which can only be sent if the request is being made from the origin related to the cookie (not cross-domain).
- **Do not use GET** requests for state-changing operations.
- Identifying Source Origin (via Origin/Referer header)
- One-time Token should be implemented for the defence of User Interaction based CSRF.
- **Secure against XSS attacks**, as any XSS can be used to defeat all CSRF mitigation techniques!

## Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

## References

- <https://www.hackingarticles.in/understanding-the-csrf-vulnerability-a-beginners-guide/>
- <https://owasp.org/www-community/attacks/csrf>
- [https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)