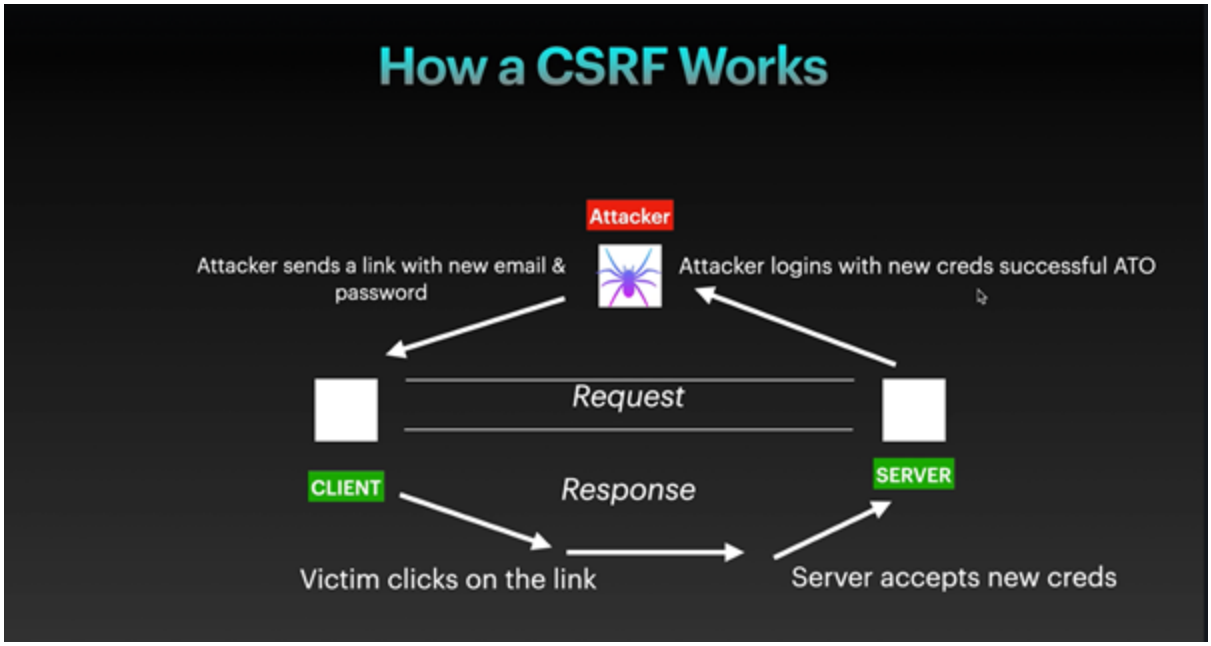
Cross Site Request Forgery (CSRF)

What is CSRF?

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.



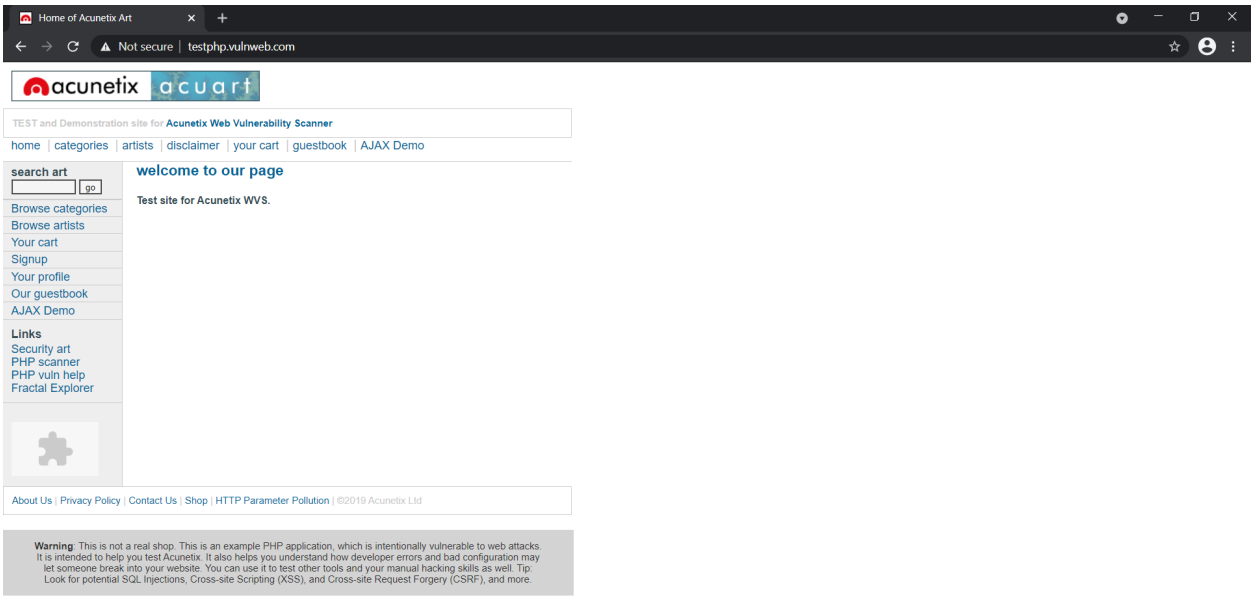
There are two main parts to execute a Cross-Site Request Forgery (CSRF) attack

1) The first part is to trick the victim into clicking a link or loading up a page. This is normally done through social engineering. By using social engineering methods attacker will lure the user to click the link.

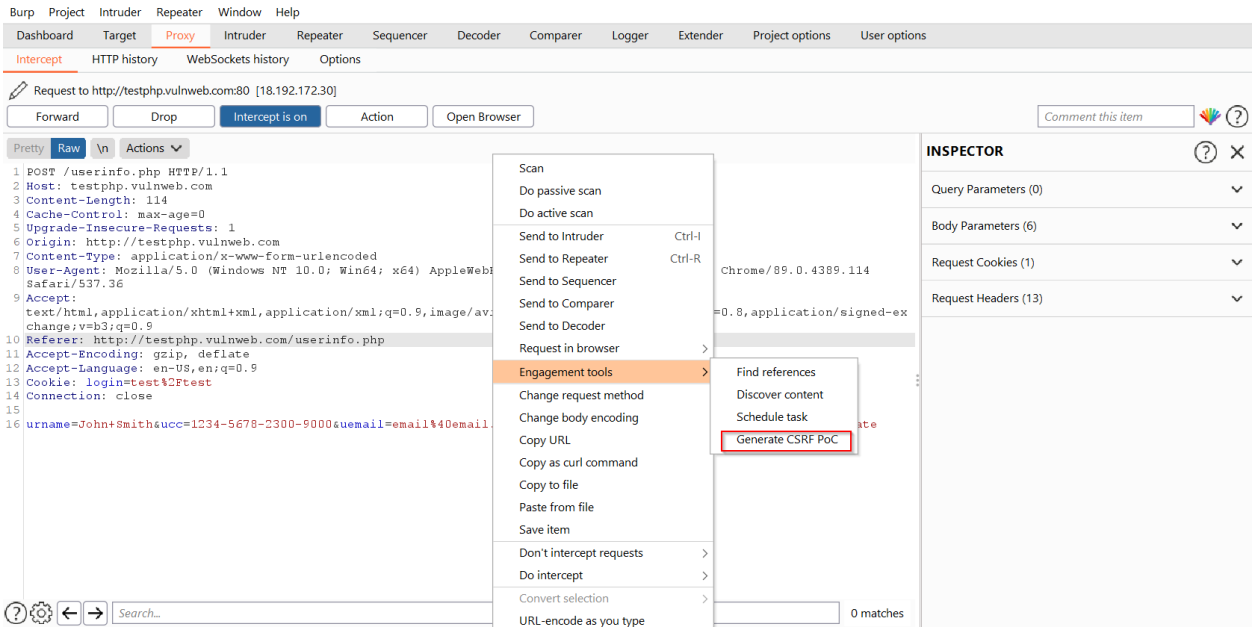
2)The second part is to send a “forged” or made up request to the victim’s browser. This link will send a legitimate-looking request to the web application. The request will be sent with the values that the attacker wants. Apart from them, this request will include any cookies that the victim has associated with that website.

Let's understand using an example

So currently I am on a vulnerable website which is: `http://testphp.vulnweb.com`



Alright now let us say is a victim account with the username `victim` and attacker account with the username `attacker` The attacker updates his account and generates the `CSRF POC`.

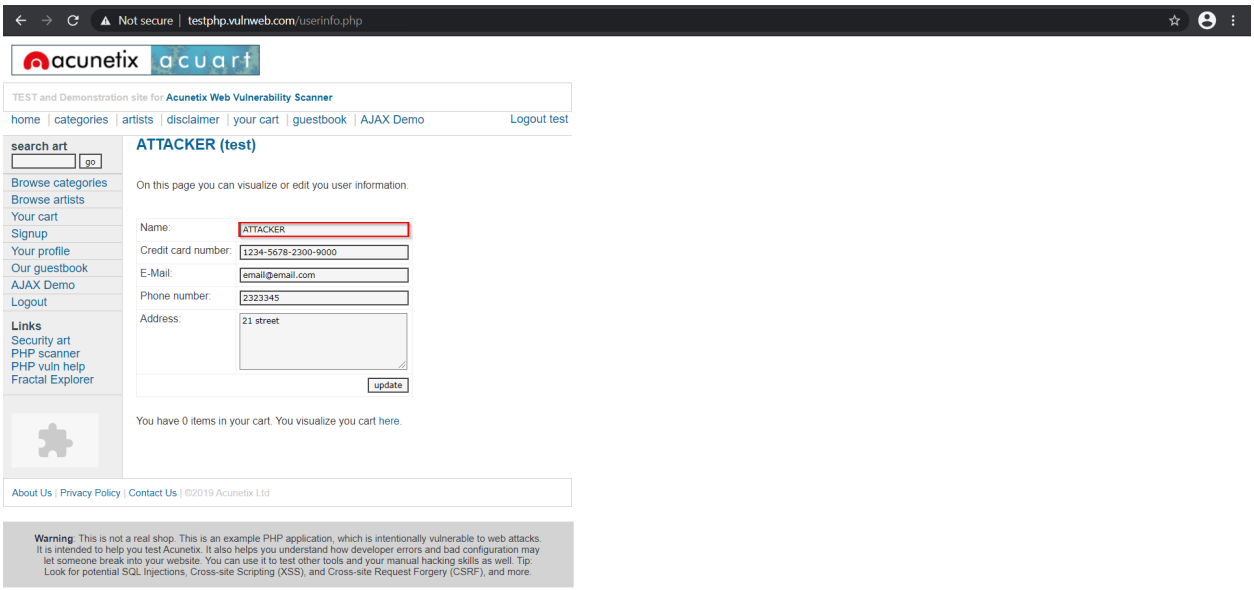


The POC looks like this:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://testphp.vulnweb.com/userinfo.php" method="POST">
      <input type="hidden" name="urname" value="John&#32;Smith" />
      <input type="hidden" name="ucc" value="1234&#45;5678&#45;2300&#45;9000" />
      <input type="hidden" name="uemail" value="email&#64;email&#46;com" />
      <input type="hidden" name="uphone" value="2323345" />
      <input type="hidden" name="uaddress" value="21&#32;street" />
      <input type="hidden" name="update" value="update" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

He then updates the POC with the details he wants to change and sends it to the victim using social engineering techniques like sending link via email.

When the victim opens the link he is redirected to the website and his details gets changed.



Boom! CSRF vulnerability was exploited!!

CSRF PoC Generator Alternative!

Burp Suite Community Edition does not come with CSRF PoC Generator an alternative tool which you can use can be found here : https://github.com/merttasci/csrf-poc-generator

Which can generate PoC for any HTTP request

Steps:

- git clone https://github.com/merttasci/csrf-poc-generator
- python -m SimpleHTTPServer 8080
- Keep the terminal running and on the browser enter: localhost:8080

Exploiting CSRF's

CSRF's can be exploited easily but there are two conditions which must be met.

1. Sending a forged request to the victim using social engineering attack.
2. The victim should click the link.

CSRF's are generally found on pages where sensitive data such as Email Id, Password, User Name can be changed

Note: CSRF on Login/Logout are not sensitive and usually Out Of Scope

The steps to exploit this vulnerability are:

- Make 2 accounts, one is of victim and another of attacker
- Sign In with attacker account and generate a malicious link also called as CSRF POC
- Send the PoC to the victim
- Sign In with the victim's account and open the link.
- If successful i.e. data changes, BOOM you proved the web application vulnerable to CSRF

Tips & Tricks

Following are some of the tips and tricks while performing CSRF attacks

- CSRF to Account Takeover: If a website is vulnerable to CSRF we can change the Email ID and/or password of the user thus performing a complete account take over of the user.

- CSRF to XSS: A simple payload of getting the cookie using XSS and passing the cookie to CSRF PoC, and then sending the malicious link to the victim can get you CSRF!
- Some web applications are vulnerable to CSRF when their request method is changed from GET to POST and vice-versa, so always try to change the request method
- Remove Tokens: Sometimes removing the token param from the PoC, can give you a valid CSRF.
- Some web applications have the CSRF Tokens as static and dynamic and hence this also can lead you to CSRF.
- Some web applications check CSRF Tokens based on entropy length. Keep the entropy length same and you win!

Severity

The severity of CSRF varies from P3 to P2 depending on what action is being performed. In cases where there is an account takeover the severity will be P2.

Impact of CSRF

In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally. Depending on the nature of the action, the attacker might be able to gain full control over the user's account. If the compromised user has a privileged role within the application, then the attacker might be able to take full control of all the application's data and functionality.

Prevention of CSRF

To remediate CSRF vulnerabilities, below are a few best practices.

- Anti-CSRF Tokens: Use a token that is associated with a particular user and can be found as a hidden value in every state changing form which is present on the web application. This token, called a CSRF Token or a Synchronizer Token
- Same Site Cookies: CSRF attacks are only possible since Cookies are always sent with any requests that are sent to a particular origin, which is related to that Cookie. Due to the nature of a CSRF attack, a flag can be set against a Cookie, tuning it into a same-site Cookie. A

same-site Cookie is a Cookie which can only be sent, if the request is being made from the same origin that is related to the Cookie being sent.

References

- CSRF by Port Swigger: https://portswigger.net/web-security/csrf
- CSRF by OWASP: https://owasp.org/www-community/attacks/csrf
- CSRF by Acunetix: https://www.acunetix.com/websitesecurity/csrf-attacks/

🧩 Lab Documentation