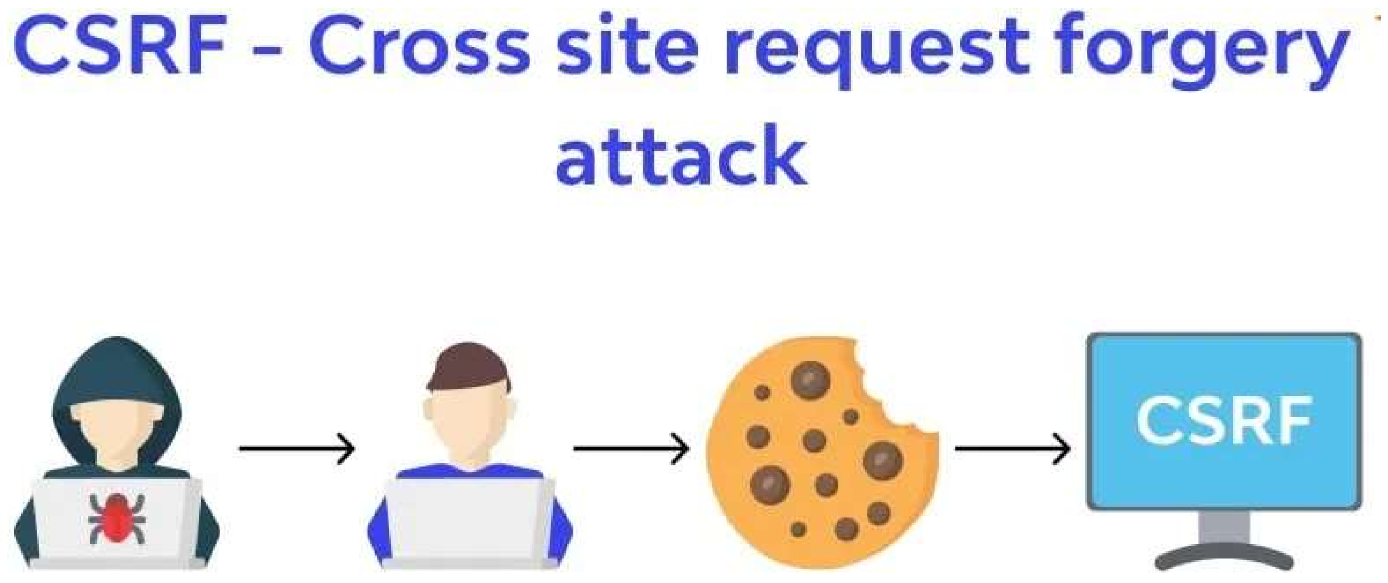
CSRF Bypass interesting techniques which can give bounty more than $3500

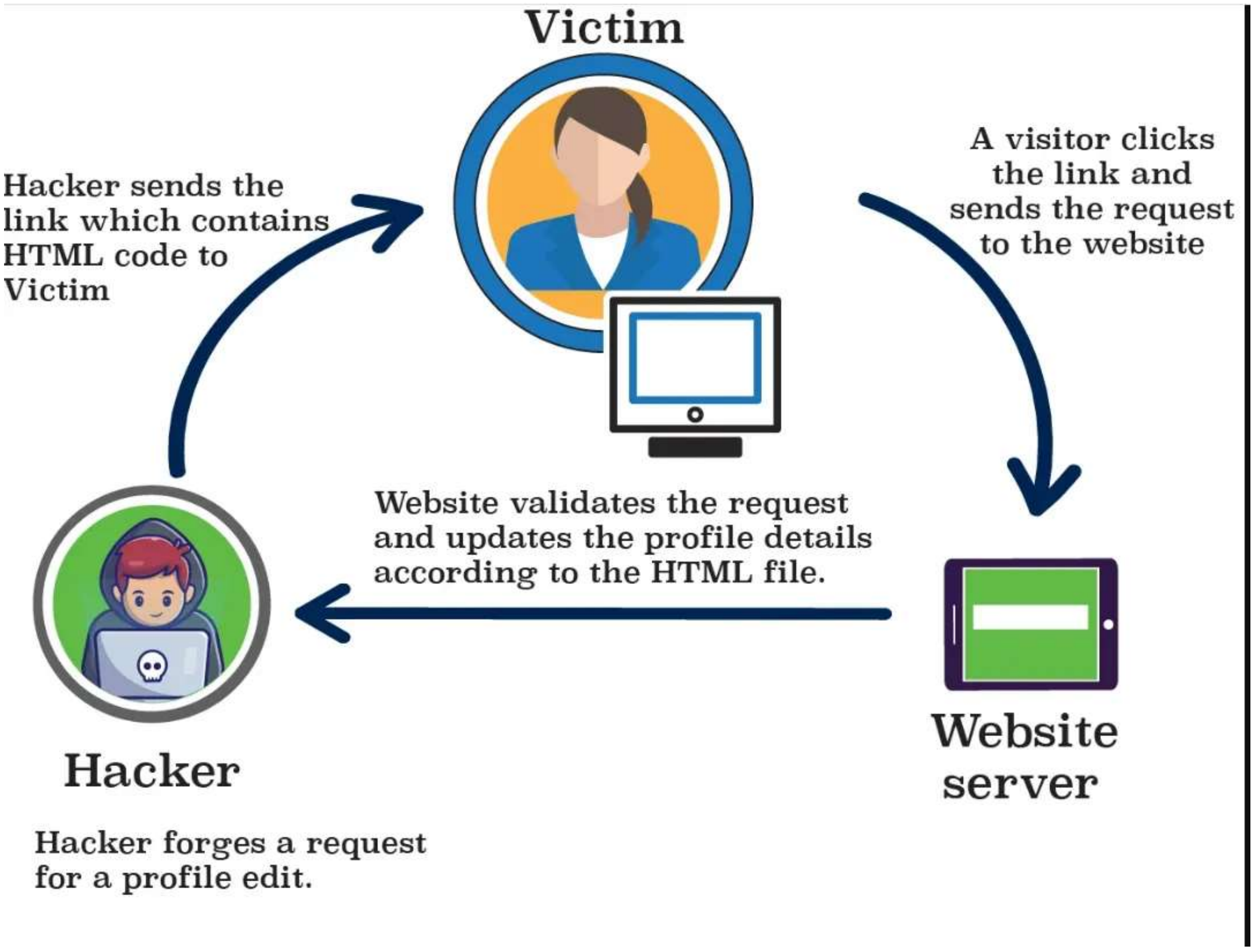


Hello, Hunters. I know You are here because you are struggling or want to advance in your career. Believe me, things take time. Be consistent, continue to learn, and never deceive yourself. If you follow these three mantras, you will undoubtedly achieve success. let read some interesting bypass method.

What exactly is Cross-Site Request Forgery?
Cross site request forgery is a web application vulnerability attack vector in which the web browser is tricked into performing an unwanted action in a susceptible application to which a user is already logged in. A properly executed CSRF attack can be devastating to both the user and the enterprise. CSRF attacks can cause email address changes, password resets, profile updates, and account takeovers. CSRFs are often carried out through the use of social engineering, such as sending the user an email or message containing our CSRF html link.

How does CSRF actually work?
A CSRF attack is conceivable under the following conditions:

An Action: A POST request or API call should be vulnerable to attack. Anything from an email change to a password reset to a profile update to the activation of 2FA could be the activity.
The programme must rely on session cookies to identify which user made the request. Session handling based on cookies. There shouldn't be any additional security measures in place to monitor user queries, such as asking for updates in secret.


Victim
Hacker sends the link which contains HTML code to Victim
A visitor clicks the link and sends the request to the website
Website validates the request and updates the profile details according to the HTML file.
Hacker
Website server
Hacker forges a request for a profile edit.

Let 's Take Below Request as an example

==================================

```
POST /profile1/edit HTTP/1.1
Host: xyz.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
```

```
Cookie: session=yunxuwoasniin425je
first_name=Paul
```

If you look in the above request , we can

Looking at the Above request, we can see that there is no token available for this post request being sent to the server. This is where the CSRF attack vectors appear. The above request essentially allows the user to edit his profile, but an attacker can now create an HTML document, as shown below.

```
<html>
  <body>
    <form action="https://redacted.com/profile1/edit” method="POST">
      <input type="hidden" name=“first_”name value=“shuttlertech” />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

The only thing left for the attacker to do is create the HTML file attack.html, host it on a domain, and distribute it to the victim by SMS or email. The user's first_name will immediately change to “shuttlertech” if he clicks the link and is already logged in to the programme.

NOTE:

Burp Suite Professional may also be used to create the HTML code previously mentioned. The majority of pentesters and bug bounty hunters utilise Burp to create their CSRF POC's by:
1. Right-click the solicitation
2. Select Engagement Tools, then create a CSRF POC.
3. Save the HTML code as Csrf.html.
4. The HTML code can be be modified whatever you like in Burp.

Impact of CSRF:

The vulnerable action will ultimately determine the impact of CSRF. If the vulnerable action is a bank transfer, the victim will incur severe financial loss because the attacker will be able to transfer funds to his account. If the activity is a password reset, the attacker will be able to instantly take over his account. As a result, the impact will always rely on the application action that is being abused, and if the susceptible action is anything critical, a patch should be implemented as soon as possible.

==

— — — — — — Bypass CSRF Protection— — — — — — —

Let's take the below request for this section:

```
POST /account2/users1/new1 HTTP/1.1
Host: xyz.com
Cookie: _ga_97CDT2HN2H=GS1.1.1623786054.8.0.1641786061.0; _ga=GA1.1.1583867464.16
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 128
Origin: https://xyz.com
Referer: https://xyz.com/account2/users1
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close
_token=ZkfcxrWQ9CeoefwlwXuIXofKB6Vnk6t7jA9n2zxG&name=none&email=tester123%40gmail
```

Bypass Method-1

After removing the CSRF token from the checking parameter, you can send the request. Many programmes enable CSRF tokens but do not check to see if the CSRF token is actually present in the parameter. As a result, the request will assume the following shape.

```
POST /account2/users1/new1 HTTP/1.1
Host: xyz.com
Cookie: _ga_97CDT2HN2H=GS1.1.1641786054.8.0.1641786061.0; _ga=GA1.1.1583877464.164
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 125
Origin: https://xyz.com
Referer: https://xyz.com/account2/users1
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close
_token=&name=none&email=tester123%40gmail.com&password=testing@123csrf&permissions
```

As you can see, the CSRF token was left out of the token parameter. If you send this request to the server and it is not validated, you have successfully bypassed the CSRF protection and are now ready to begin a CSRF attack.

Bypass Method-2

You can use another user's CSRF token, which only requires that you make a new account, intercept the same request — the profile edit request — and copy the CSRF token from it. Then, you can paste that CSRF token into the first request to see if the server is actually validating whether the CSRF token actually belongs to that specific user or not.

Suppose User 1 CSRF Token is : `KQfcxrWQ9CeoefwlwXuIXofKB6Vnk6t7jA9n2zxG`

Suppose User 2 CSRF Token is : `SE49jfXwQmExuz2SJGF91PK5WfMVu5sBOGgzqqvn`

You must now copy User 2's CSRF token and paste it in User 1's request to edit their profile before sending the request and checking to see if the server validates it. In this situation, you have successfully avoided the CSRF safeguards and can proceed with your CSRF assaults.

Bypass Method-3

Add random text in the CSRF token and check if the server is validating it or not. So all you have to do is this :

Real CSRF Token : `KQfcxrWQ9CeoefwlwXuIXofKB6Vnk6t7jA9n2zxG`

Malformed CSRF Token : `KQfcxrWQ9CeoefwlwXuIXofKB6Vnk6t7jA9n2zxGrandom`

Now all you have to do is include the faulty cstf token in your request and wait for server validation. If you're lucky and the server doesn't validate, you've successfully circumvented csrf protection and can carry out your CSRF assaults.

Bypass Method-4

You can see that most of the Profile edits, password resets and 2FA enabling are POST requests but what we are going to do in this bypass is basically changing the request type to GET and removing the CSRF token parameter and forward the request.

So, the request should ultimately like the below:

```
GET /account2/users1/new?name=none&email=tester123%40gmail.com&password=testing@1
Host: xyz.com
Cookie: _ga_97CDT2HN2H=GS1.1.1641786054.8.0.1641786061.0; _ga=GA1.1.1583877464.16
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Origin: https://xyz.com
Referer: https://xyz.com/account2/users1
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close
```

As you can see, I removed the token parameter from the preceding call and changed the adding user's POST request to GET. If the server administrator hasn't tested them with multiple request methods, this bypass should work and allow you to get around any installed CSRF defences.

Bypass Method-5

Many Web applications use a Static and Dynamic component CSRF token, as shown below.

CSRF Token : `KscgdrWQ9CeoefwlwXuIXofKB6Vnk6t7jA9n2zxG`

The first 20 characters of this CSRF token are static, which means they are the same for every user registered with the Web application, and the latter 20 characters are dynamic, which means they vary based on the user. As a result, you can use random text for the dynamic 20 characters while keeping the CSRF token's static characters the same. If the server accepts the CSRF token, you have successfully avoided CSRF protection.

==

Bonus Tips

Check the token's unpredictability, automate the process with BURP, assess whether the token is weak or not, and try to crack it. Check to determine if the CSRF token is merely a standard hash token, such as MD5, and if so, replace the token with a new one made using that hash method. Switch the User-Agent to Mobile to see if the request is permitted.

==

Thank you for reading !! hope you get to learn some tricks.

Subscribe to the Shuttlertech YouTube channel for more of this type of content and to watch live POCs & To advance your career connect with me 1:1 over topmate .
I will ensure you that I will write more interesting and knowledge-sharing writeups, to encourage me to follow me on medium and click the clap icon.

Disclaimer: My write-up comes from my own achievements & Some time from different Learning platforms Do not use this methodology without concern for the company. I am just sharing this for learning purposes.