HTB Academy - Using Web Proxies

Ernesto Ramos (h4xthan)

June 27, 2025

Overview

This document contains the solutions and methodologies used to solve the four challenge exercises provided in the module. The aim is to demonstrate the understanding of concepts such as DOM manipulation, encoding/decoding cookies, fuzzing, and HTTP request analysis.

1 Exercise 1: Disabled Button on /lucky.php

Challenge

The page `/lucky.php` has a button that appears to be disabled. Enable the button and click it to get the flag.

Approach

Upon inspecting the page using DevTools, a `disabled` attribute was found in the button's HTML:

```
<button class="btn␣block-cube␣block-cube-hover" id="submit" type="submit"
   formmethod="post" name="getflag" value="true" disabled="">
```

Removing the `disabled` attribute allowed interaction with the button. Initially, nothing happened when clicking. A POST request was intercepted in Burp Suite:

```
POST /lucky.php HTTP/1.1
Host: 94.237.48.12:33867
Content-Type: application/x-www-form-urlencoded
...
getflag=true
```

Upon retrying later with the same steps, the flag appeared.

Flag

`HTB{d154bl3d_bu770n5_w0n7_570p_m3}`

2 Exercise 2: Decoding a Multi-Encoded Cookie

Challenge

The page `/admin.php` uses a cookie that is encoded multiple times. Decode it until a 31-character value appears.

Approach

The cookie was first decoded from ASCII Hex, then Base64, revealing the value:

```
3dac93b8cd250aa8c1a36fffc79a17a
```

Answer

`3dac93b8cd250aa8c1a36fffc79a17a`

3 Exercise 3: Fuzzing the Last Character of a Hash

Challenge

After decoding the cookie, the 31-character string appears to be an incomplete MD5 hash. Fuzz the final character using all alphanumeric values. Encode each request using the same encoding methods previously identified. Use the `alphanum-case.txt` wordlist.

Approach

Using Burp Suite's Intruder, the following steps were taken:

1. Prefix the decoded cookie value.
2. Use the `alphanum-case.txt` wordlist as payload.
3. Use Burp's `Payload Processing` to encode each guess first as Base64, then as ASCII Hex.
4. Replace the cookie value dynamically using Intruder's markers.

Initially, the fuzzing was unsuccessful because only the suffix was marked. By switching the entire cookie string to be fuzzed (with Intruder markers), encoding and submission worked correctly.

Flag

`HTB{burp_1n7rud3r_n1nj4!}`

4 Exercise 4: ColdFusion Directory Enumeration

Challenge

Using the `coldfusion_locale_traversal` module in Metasploit, capture the HTTP request. Determine which directory is being called in `/XXXXX/administrator/...`

Approach

Configured Metasploit with proxy settings to route through Burp Suite. Upon execution, the following request was intercepted:

```
/CFIDE/administrator/...
```

Answer

`CFIDE`

Conclusion

This challenge set was an excellent exercise in client-side inspection, cookie manipulation, fuzzing with Burp Suite, and understanding of how Metasploit modules make requests. While some parts were initially confusing, a methodical and persistent approach led to success.