Lab 4: Authentication Vulnerabilities (multi-factor authentication)

Challenge 1: 2FA simple bypass

Part 1: Description and Objective

- **Objective:** The goal of this lab is to bypass the Two-Factor Authentication (2FA) mechanism and access the account of the victim user, **carlos**.
- **Vulnerability:** The application likely suffers from a forced browsing vulnerability. It may not strictly enforce the 2FA check before allowing access to authenticated pages (like the "My Account" page).
- Credentials:
    - Your credentials: wiener / peter
    - Victim credentials: carlos / montoya

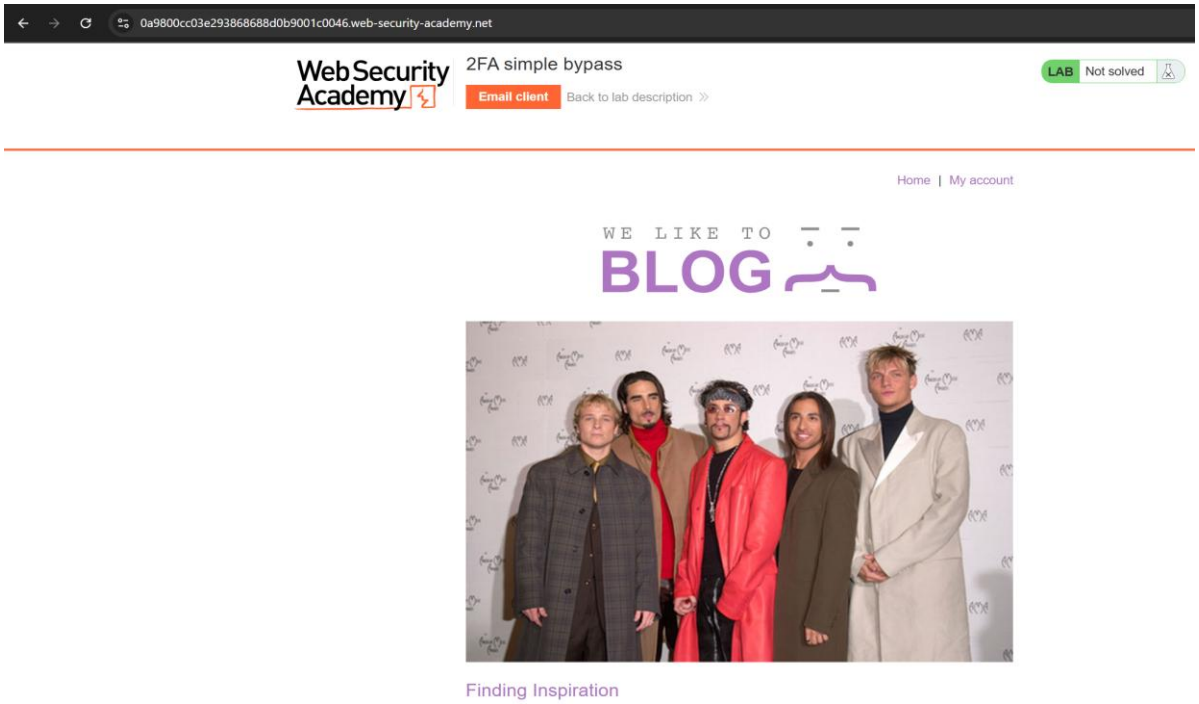
Part 2: Solution Overview

1. Log in to your own account. Your 2FA verification code will be sent to you by email. Click the **Email client** button to access your emails.
2. Go to your account page and make a note of the URL.
3. Log out of your account.
4. Log in using the victim's credentials.
5. When prompted for the verification code, manually change the URL to navigate to `/my-account`. The lab is solved when the page loads.

Part 3: Step-by-Step Implementation

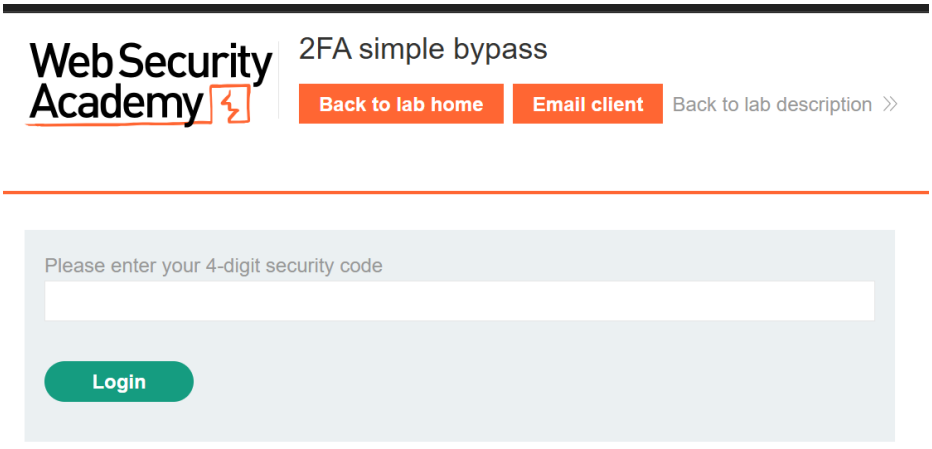
Step 1: Access the Lab

- **Action:** Navigate to the lab "2FA simple bypass" on PortSwigger Academy.
- **Execution:** Click the **"ACCESS THE LAB"** button to start the instance.

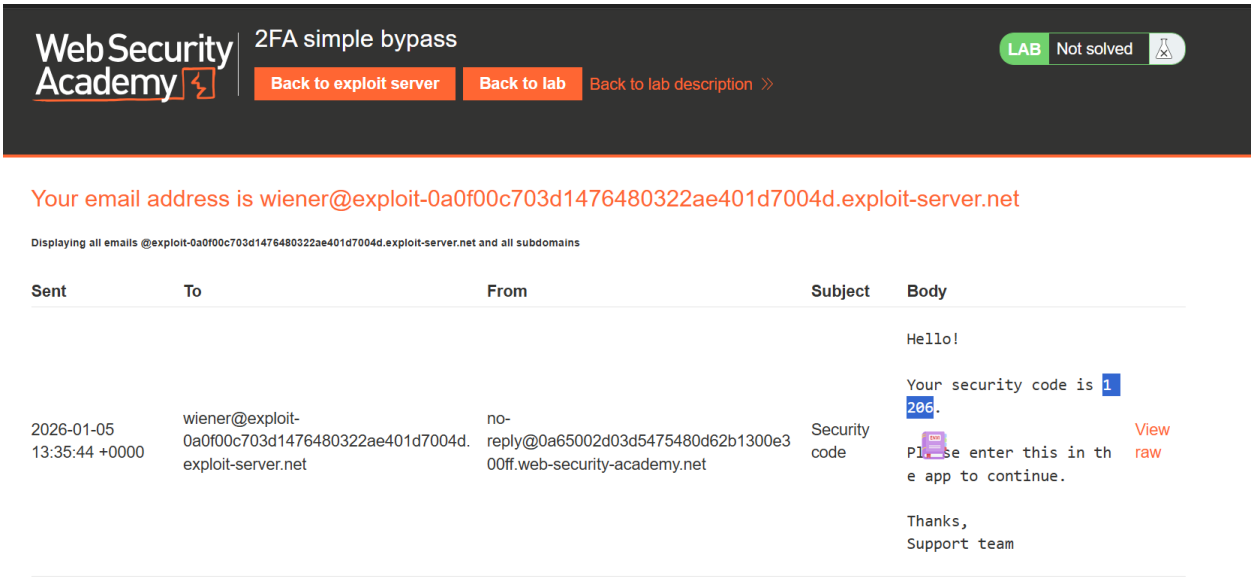


Step 2: Analyze the Login Flow (Using Own Account)

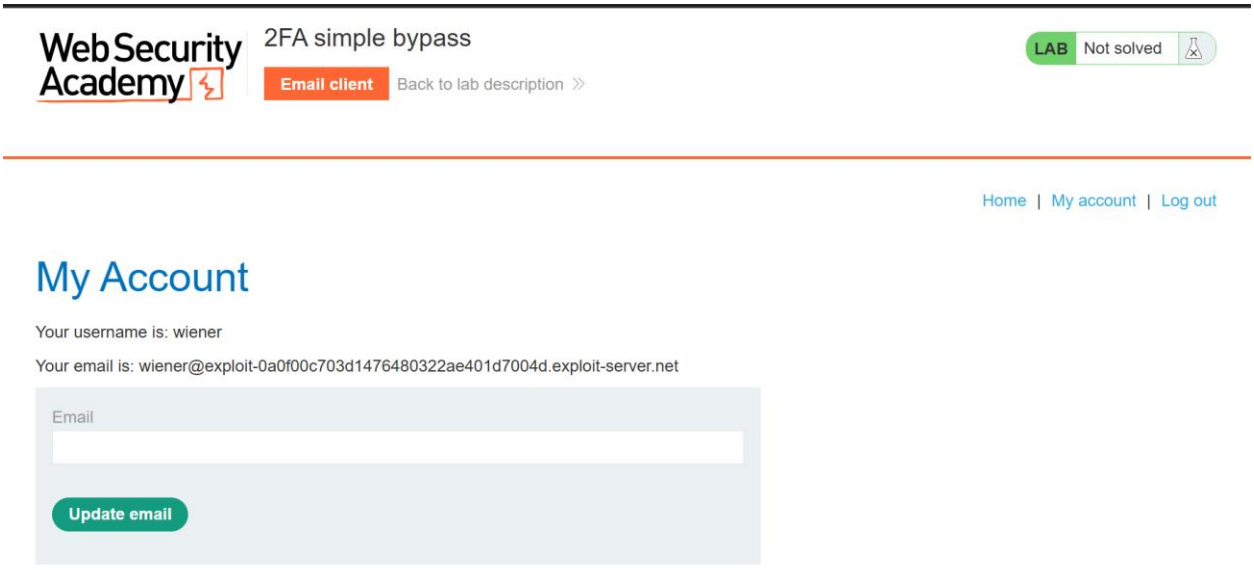
- **Action:** Log in with the provided valid credentials (wiener / peter) to observe the URL structure after a successful 2FA verification.
- Execution:
    1. Click the **"My account"** link in the top right corner.
    2. Enter the username **wiener** and password **peter**, then click **"Log in"**.
    3. When prompted for the 4-digit security code, click the **"Email client"** button (located in the top lab banner) to open the email simulation in a new tab.



4. Copy the 4-digit code from the email body.



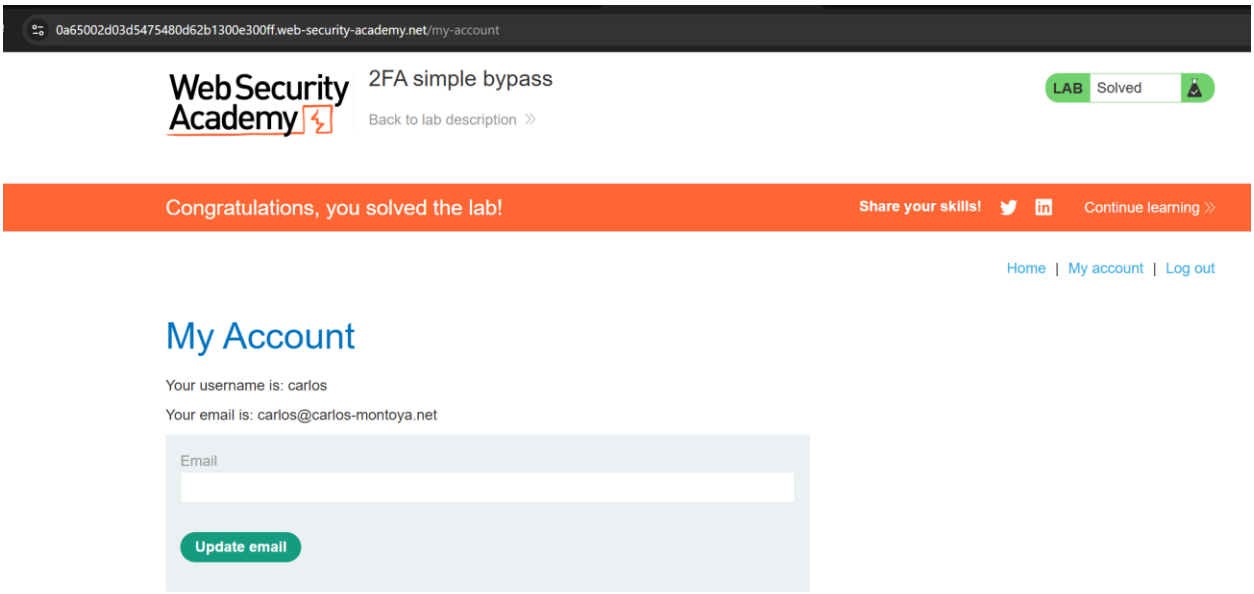
5. Return to the login page, enter the code, and click **"Log in"**.
6. Observation: Upon successful login, observe the URL in the browser's address bar. It should be /my-account. This confirms the target URL we need to access.



Step 3: Bypass 2FA on Victim's Account

- **Action:** Exploit the forced browsing vulnerability to access the victim's account without the 2FA code.
- Execution:
    1. Log out of the wiener account.

2. Go to the login page and enter the victim's credentials:
    a. Username: **carlos**
    b. Password: **Montoya**
3. Click **"Log in"**. You will be redirected to the 2FA verification page (/login2).
4. **Bypass:** Instead of entering a code (which you do not have), manually change the URL in the browser's address bar from /login2 to **/my-account**.
5. Press **Enter**.



Challenge 2: 2FA broken logic

Part 1: Description and Objective

- **Objective:** Access the account of the victim user, **carlos**, who has 2FA enabled.
- **Vulnerability:** The application's 2FA logic is flawed. While the login page might have rate limiting, the 2FA verification page often lacks it. Additionally, the mechanism for verifying the code relies on a predictable or manipulatable session identifier (like a cookie) that determines *which* user is being verified.
- Credentials:
    - Your credentials: wiener / peter
    - Victim credentials: carlos / Montoya

Part 2: Solution Overview

The strategy involves a brute-force attack on the 2FA code, made possible by a logic flaw:

1. With Burp running, log in to your own account and investigate the 2FA verification process. Notice that in the POST /login2 request, the verify parameter is used to determine which user's account is being accessed.

2. Log out of your account.
3. Send the `GET /login2` request to Burp Repeater. Change the value of the `verify` parameter to `carlos` and send the request. This ensures that a temporary 2FA code is generated for Carlos.
4. Go to the login page and enter your username and password. Then, submit an invalid 2FA code.
5. Send the `POST /login2` request to Burp Intruder.
6. In Burp Intruder, set the `verify` parameter to `carlos` and add a payload position to the `mfa-code` parameter. Brute-force the verification code.
7. Load the 302 response in the browser.
8. Click **My account** to solve the lab.

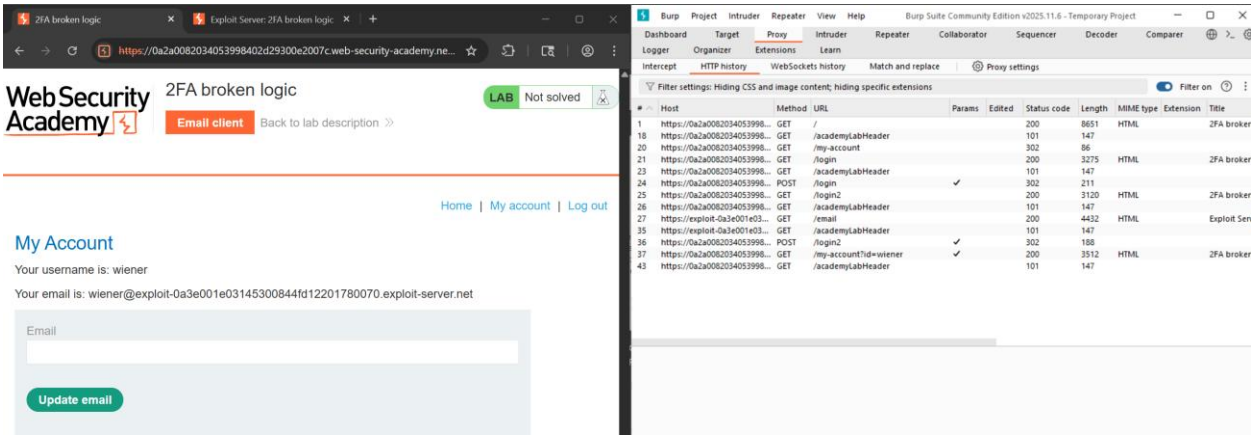
Part 3: Step-by-Step Implementation

Step 1: Access the Lab

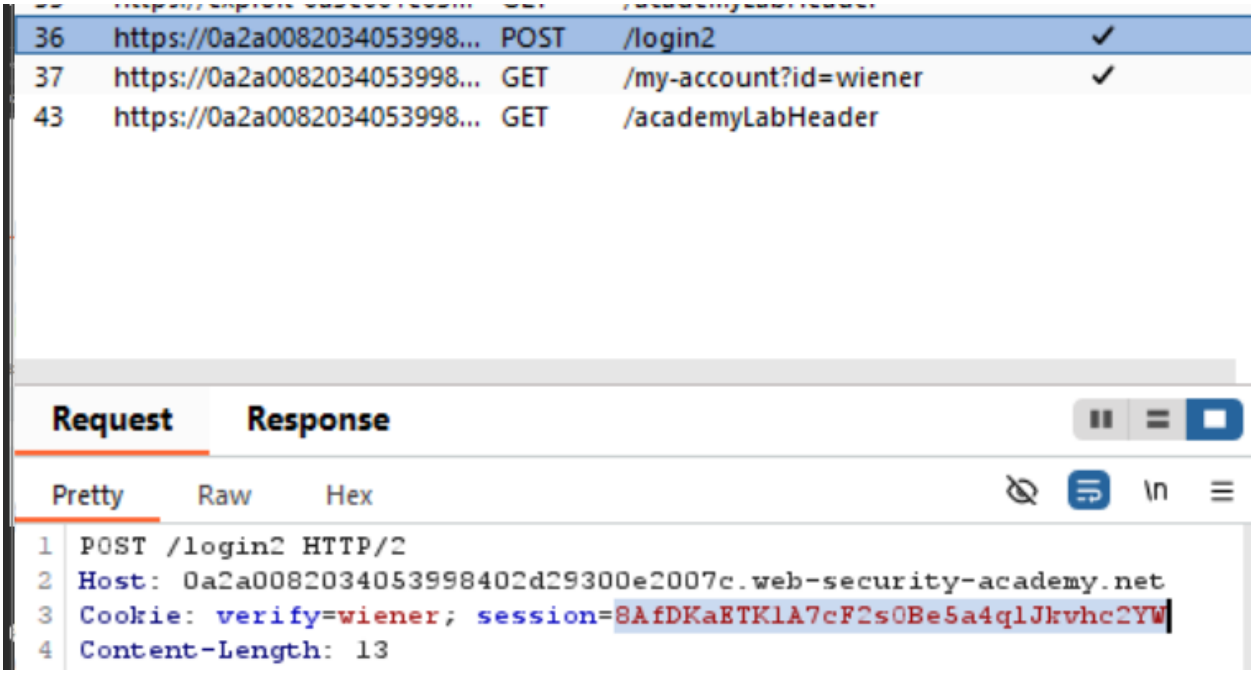
- **Action:** Navigate to the lab "2FA broken logic" on PortSwigger Academy.
- **Execution:** Click **"ACCESS THE LAB"** to start the instance.

Step 2: Analyze the 2FA Request (Using Own Account)

- **Action:** Capture a valid 2FA verification request to understand the parameters.
- Execution:
    1. Log in as **wiener** / **peter**.
    2. When prompted for the code, check the **Email client**, get the code, and enter it
    3. **Important:** Before clicking "Log in" (or immediately after), check **Burp Suite HTTP History**.

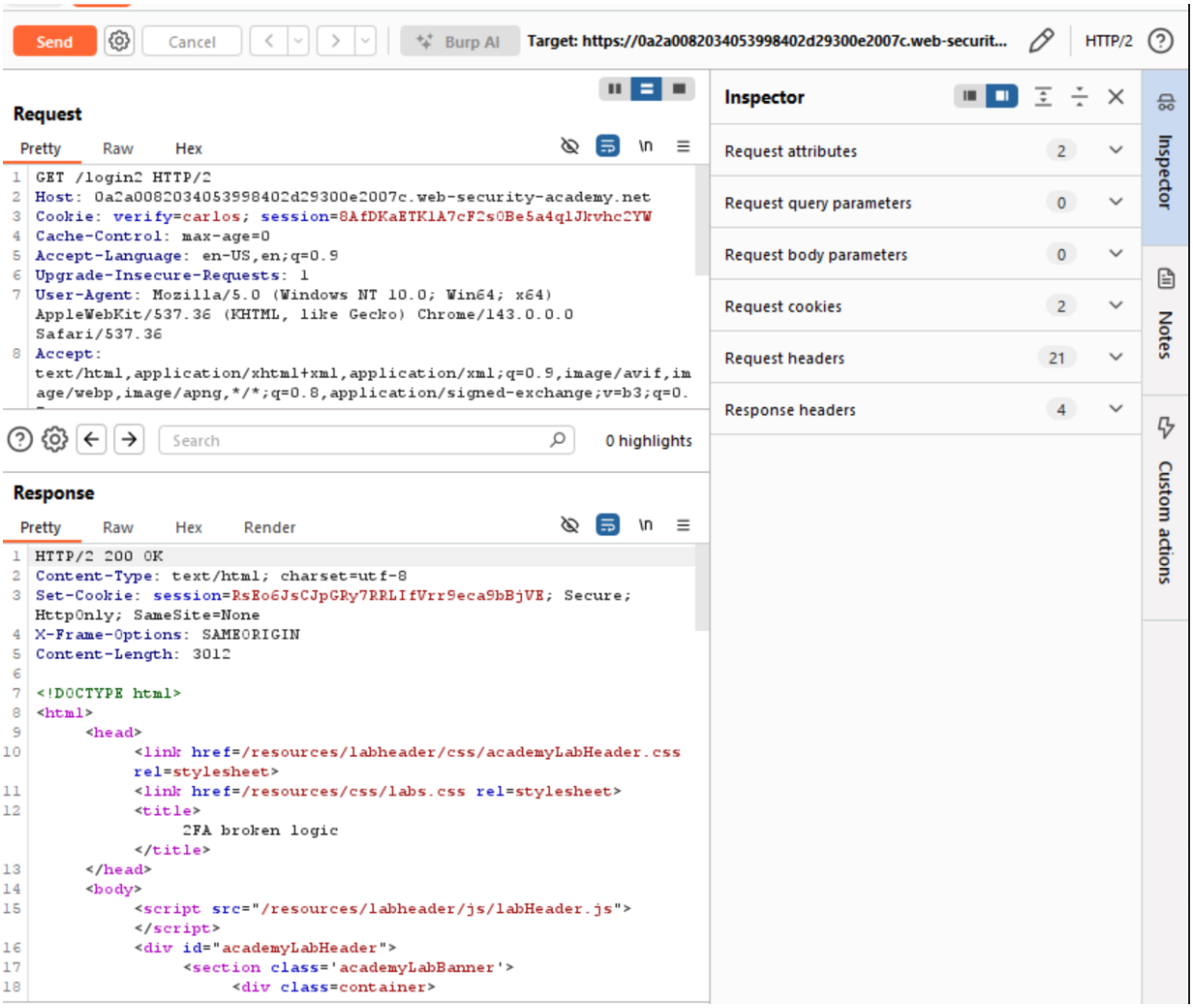


4. Find the POST /login2 request (the one sending the mfa-code).
5. **Observation: Inspect the Cookie header. It contains a parameter verify=wiener.**
6. **Conclusion:** The application uses this cookie to determine which user's 2FA code is being verified. If we can generate a 2FA session for Carlos and manipulate this cookie (or just use Carlos's cookie), we can brute-force the code.



Step 3: Trigger 2FA Code Generation for Carlos

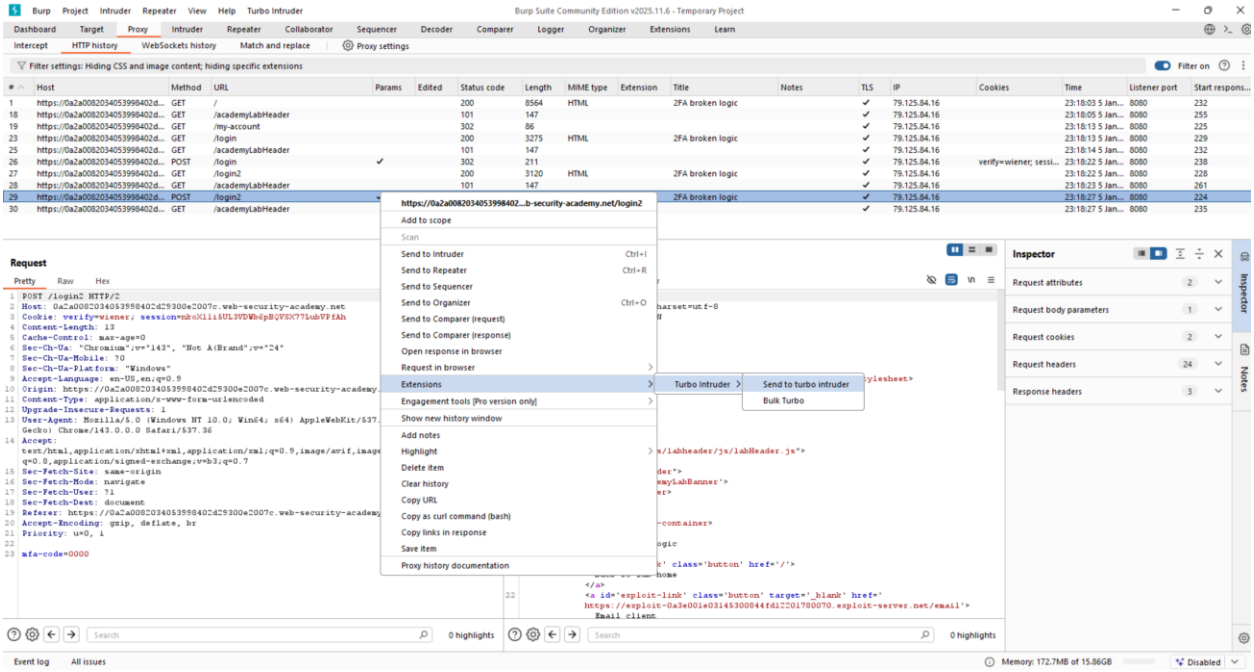
- **Action:** Use Burp Repeater to manipulate the verification cookie and force the server to generate a 2FA code for the victim (carlos).
- Execution:
    1. Log in as **wiener** / **peter**. The 2FA page loads.
    2. In **Burp Suite > Proxy > HTTP history**, locate the GET /login2 request (the request that loads the 2FA page).
    3. Right-click this request -> **Send to Repeater**.
    4. In the **Repeater** tab:
        a. Locate the request header: Cookie: verify=wiener...
        b. Change it to: Cookie: verify=carlos...
    5. Click **Send**.
    6. **Observation:** The response should be **200 OK**. This action triggers the server to generate a new 4-digit code for carlos (internally).

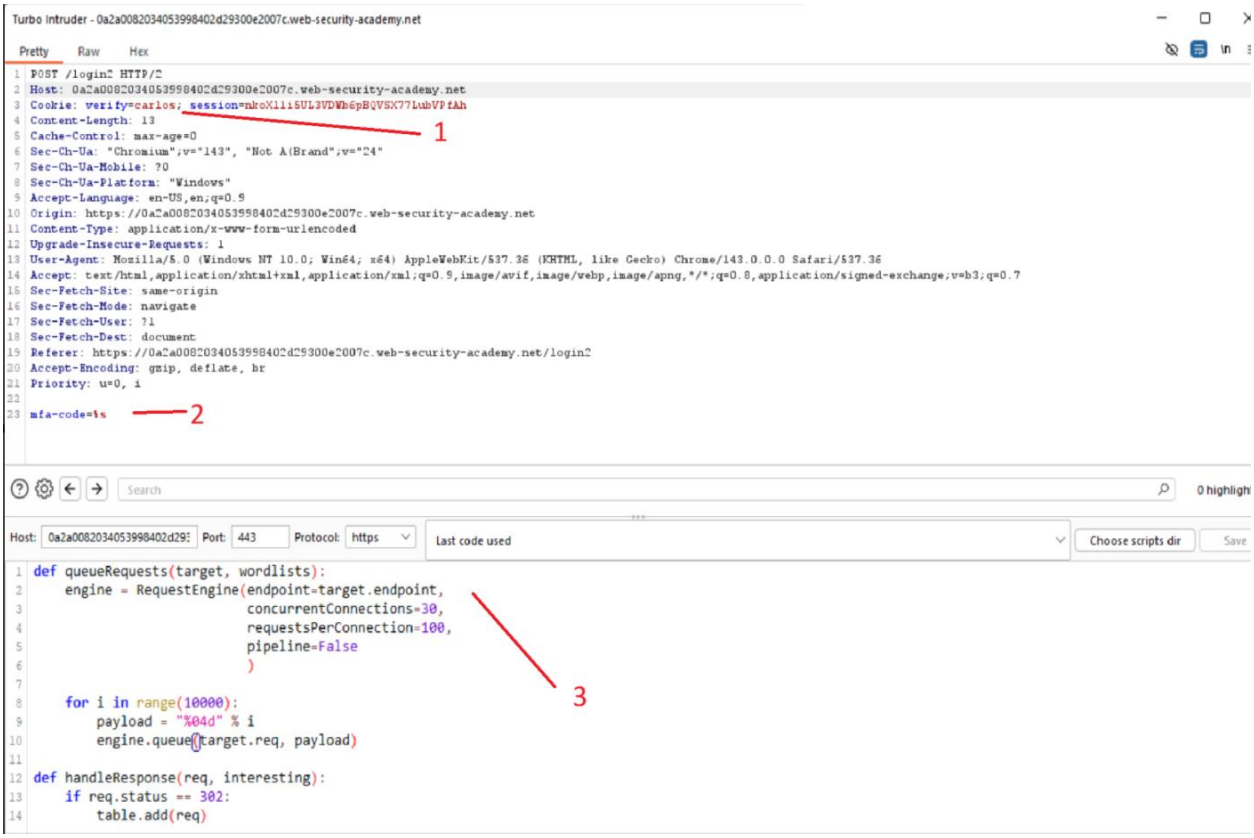


Step 4: Launch Attack and Identify the 2FA Code

- **Action:** Execute the brute-force attack and identify the valid 2FA code based on the HTTP status code.
- Execution:

1. Initiate the attack (using Turbo Intruder or Burp Intruder).



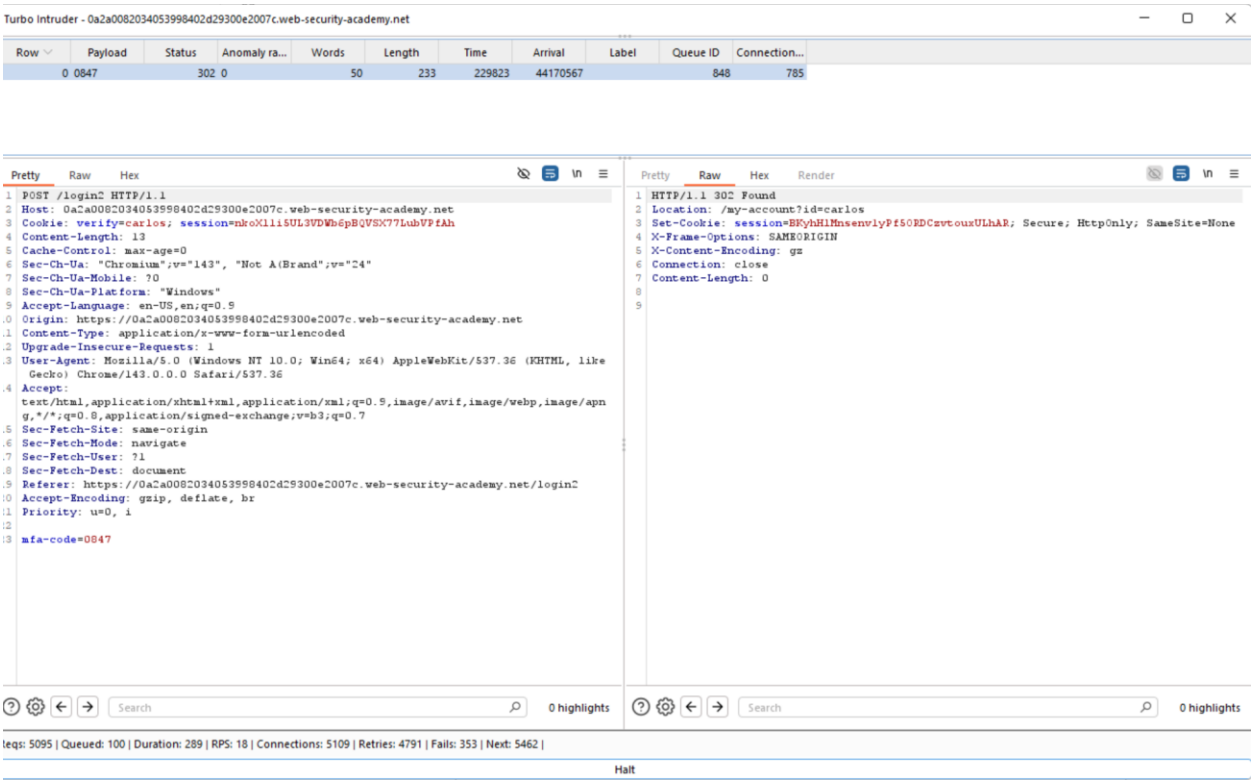


2. **Analysis:** Monitor the results table for any request returning a **302 Found** status code.

3. **Observation:** The payload **0847** triggered a **302 Found** response.

4. Critical Action: Inspect the Response headers of this successful request. Locate and copy the value of the new session cookie (e.g., BKyhHl...).

5. Conclusion: The server has accepted the code 0847 and issued a valid session cookie for Carlos.



Step 5: Session Hijacking (Final Verification)

- **Action:** Manually inject the captured session cookie into the browser to bypass the login prompt and access the victim's account.
- **Reasoning:** The 2FA code is only valid for the specific session used by the attack tool. To access the account in the browser, we must "hijack" the valid session by replacing the browser's cookie.
- Execution:

  1. Open the browser's **Developer Tools (F12)** and navigate to the **Application > Cookies** tab.

  2. Locate the **session** cookie.

  3. **Modification:** Replace its value with the authenticated session string copied from Step 5.

  4. Update the verify cookie to carlos.

5. Reload the page or navigate to /my-account.

6. **Result:** The application authenticates the request using the hijacked cookie, granting access to Carlos's account. The lab status updates to **"Solved"**.