SCENARIO

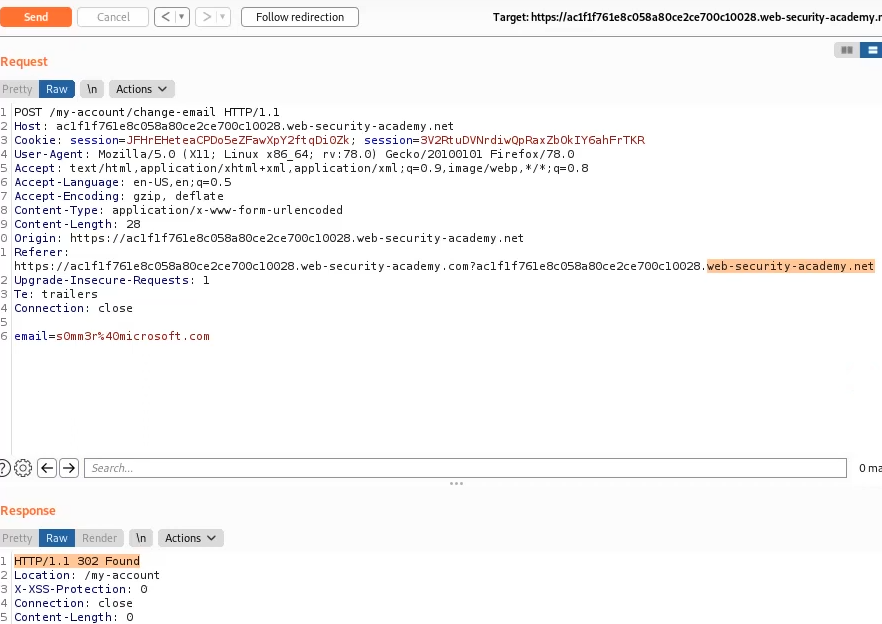
The lab's email modification feature is found to have a CSRF vulnerability. It relies on the unsafe "double submit" method for CSRF mitigation. The flaw allows external entities to craft malicious links that, when clicked by an authenticated user, can change that user's email address without their knowledge.

**PROCEDURE**

1. Initialize Burp's browser and sign in to our designated account.
2. Utilize the "Update email" form and submit the data. Subsequently, identify the resulting network request within Burp's Proxy history.
3. Forward this request to the Burp Repeater tool and observe how the csrf body parameter is matched against the csrf cookie for validation.
4. Execute a search operation, direct its request to the Burp Repeater, and notice that the entered search term is reflected in the Set-Cookie header. The lack of CSRF protection for this search operation allows us to exploit it to introduce cookies into an unsuspecting user's browser.
5. Design a URL that takes advantage of this gap to implant a counterfeit csrf cookie into the victim's system: /?search=test%0d%0aSet-Cookie:%20csrf=fake%3b%20SameSite=None.
6. Craft and host a demonstration exploit using insights from our analysis of the CSRF flaw, ensuring the CSRF token is labeled "fake". The foundation for this exploit is the email modification request.
7. Omit the auto-submit script segment and instead incorporate the subsequent code to introduce the cookie and subsequently activate the form: <img src="https://YOUR-LAB-ID.web-security-academy.net/?search=test%0d%0aSet-Cookie:%20csrf=fake%3b%20SameSite=None" onerror="document.forms[0].submit();"/>.
8. Modify the email address within our exploit, ensuring it is distinct from the original.
9. Finally, save and execute the exploit by clicking "Deliver to victim".

**PAYLOAD**

<img src="https://YOUR-LAB-ID.web-security-academy.net/?search=test%0d%0aSet-Cookie:%20csrf=fake%3b%20SameSite=None" onerror="document.forms[0].submit();"/>

**PROOF OF CONCEPT**

**REMEDIATION**

1. Adopt the secure method of utilizing anti-CSRF tokens in all sensitive forms.
2. Steer clear of reflecting user inputs directly in Set-Cookie headers.
3. Implement a robust Content Security Policy (CSP) to restrict the domains that can be loaded and executed as part of the web application.
4. Offer proper training to developers regarding the hazards linked to CSRF attacks and suitable protection measures.
5. Habitually scrutinize and update software components to shield against familiar vulnerabilities.