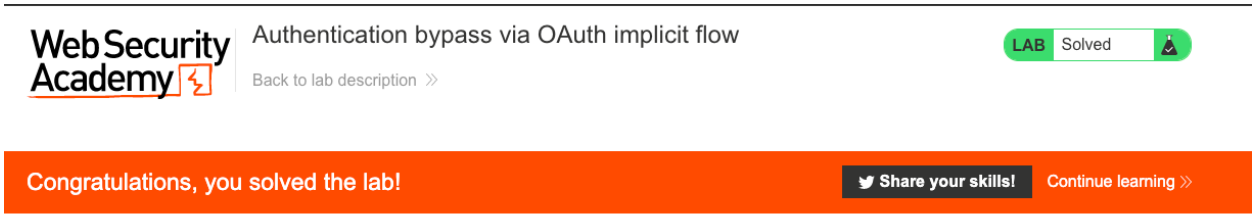
Final Project

Recording level completion

Bradley Thompson | CS595 Web and Cloud Security | Winter 2022

Level 1: Authentication bypass via OAuth implicit flow

Solved with previous solution from lab 1.3



Level 2: Forced OAuth profile linking

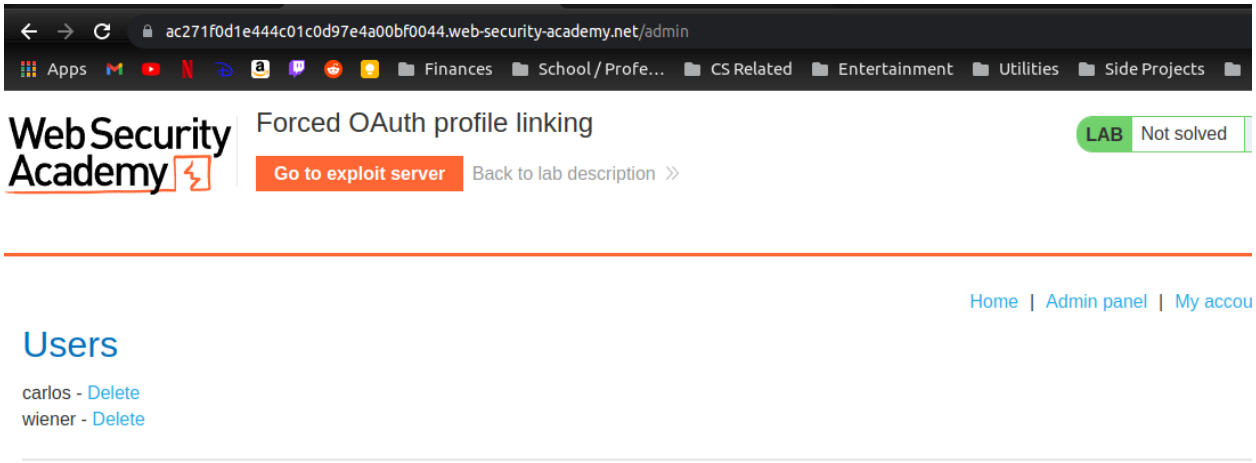
Go through OAuth flow programmatically so that we can intercept the final redirect after the oauth confirmation page. This has the OAuth code in the Location header. Upload that to the exploit server so that the admin validates our next social login attempt.

Code that grabs that token from confirm response headers.

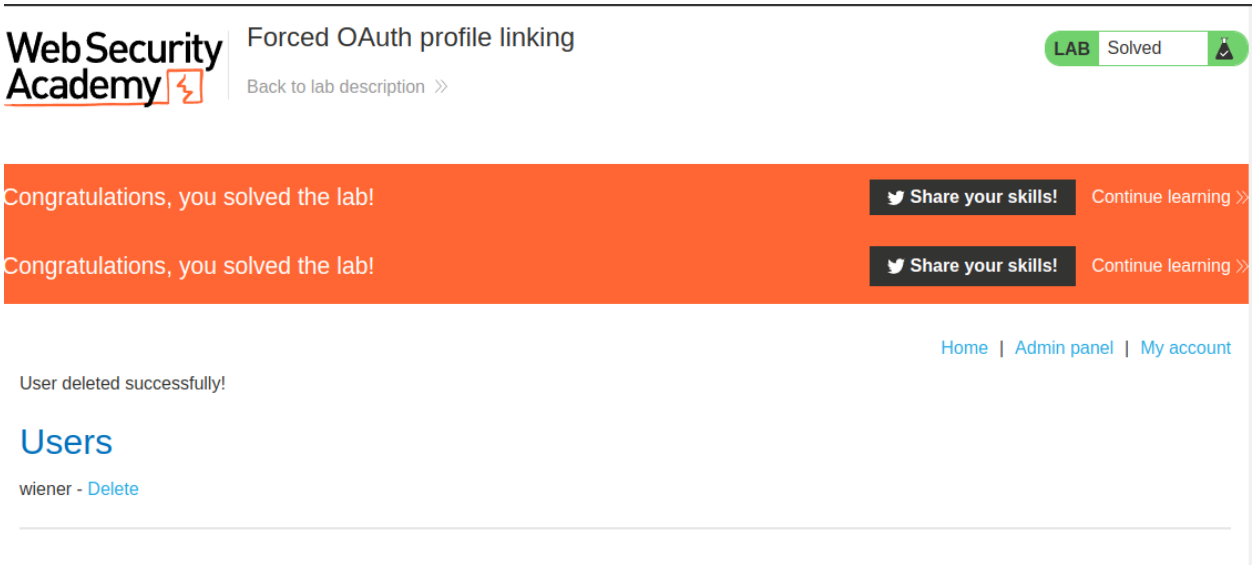
```
# Find validated oauth token in attempted redirect response
oauth_token_get_url = resp.headers['Location']
print(f'Confirm submission attempting to redirect to: {oauth_token_get_url}')
oauth_token = searchResponse(s.get(oauth_token_get_url, allow_redirects=False), 'a').get('href').split('code=')[1]
print(f'Found valid OAuth token! Token: {oauth_token}')

# Make admin validate with this token
exploit_payload = f'''
    <iframe src="https://{site}/oauth-linking?code={oauth_token}"></iframe>
'''
resp = uploadExploit(s, site, exploit_payload, 'DELIVER_TO_VICTIM')
if resp.status_code == 200:
    print("Payload delivered to victim, login now.")
# Now, in UI should be able to click "Log in with social media" and get logged in as admin.
```

Because we stop the redirect, the code remains valid for the next login attempt.
After clicking ‘Login with Social Media’, can access Admin Panel:

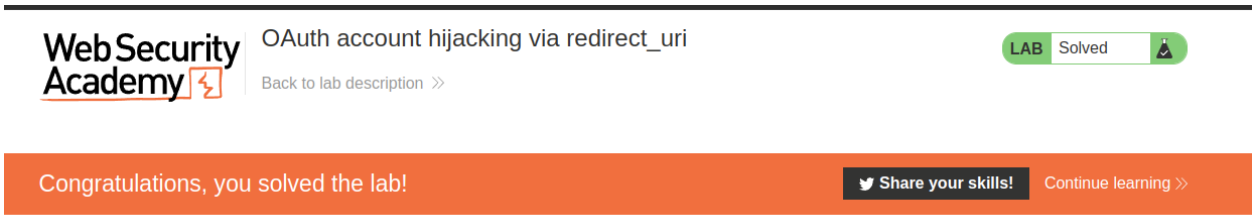


Success



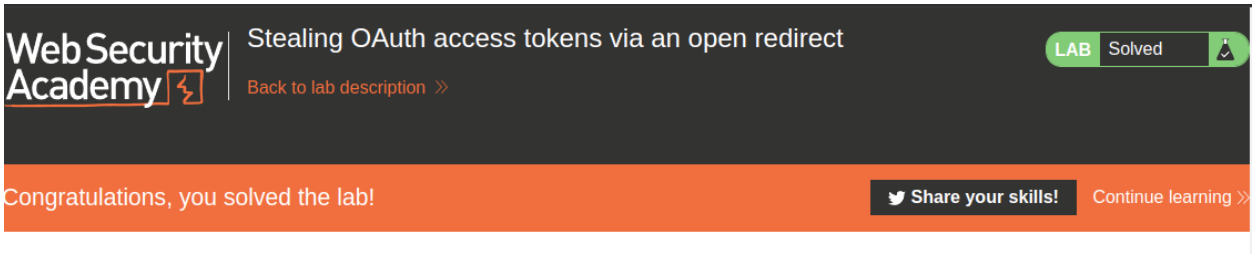
Level 3: OAuth account hijacking via redirect_uri

Observed that client provides the redirect_uri to use following the completion of the OAuth flow. Send that redirect to the exploit server to find the admin's personal oauth code, then go directly to the oauth-callback w/ that code to login as admin.



Level 4: Stealing OAuth access tokens via an open redirect

Used the open redirect in post page to trick the client to redirect to exploit server, leave access token in access log.

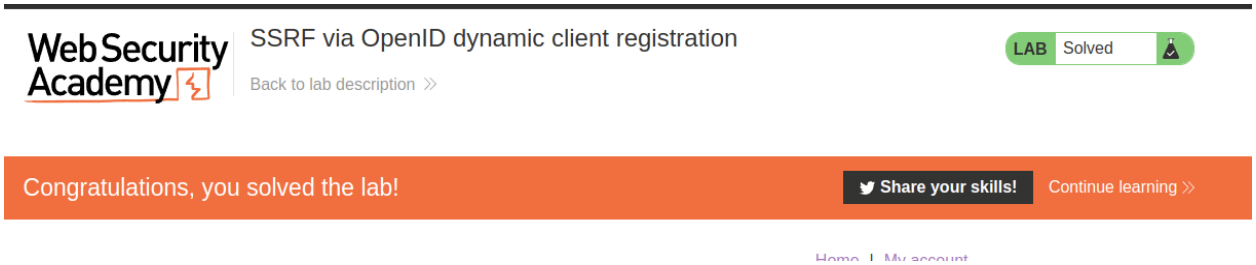


Level 5: SSRF via OpenID dynamic client registration

Take advantage of the fact that the OpenID client registration process causes the server to try and access a logo via a ‘logo_uri’ provided in client configuration. No restriction on the origin of that logo_uri, so I got the server to access the provided URL in the lab description.

From that, I was able to find the admin’s client id. Using that to access the logo, which is a json blob full of admin meta data instead of an image file now, thanks to the aforementioned steps.

Grab SecretAccessKey from that pages response and submit to solve level



Level 6: Stealing OAuth access tokens via a proxy page

Altered level 4 solution to take advantage of the fact that the comment form sends message events to any origin. Setup the exploit payload such that those messages are caught by an event listener and sent to the exploit server. Grabbed access token from the exploit server logs.

Web Security Academy

Stealing OAuth access tokens via a proxy page

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning »

Home | My account