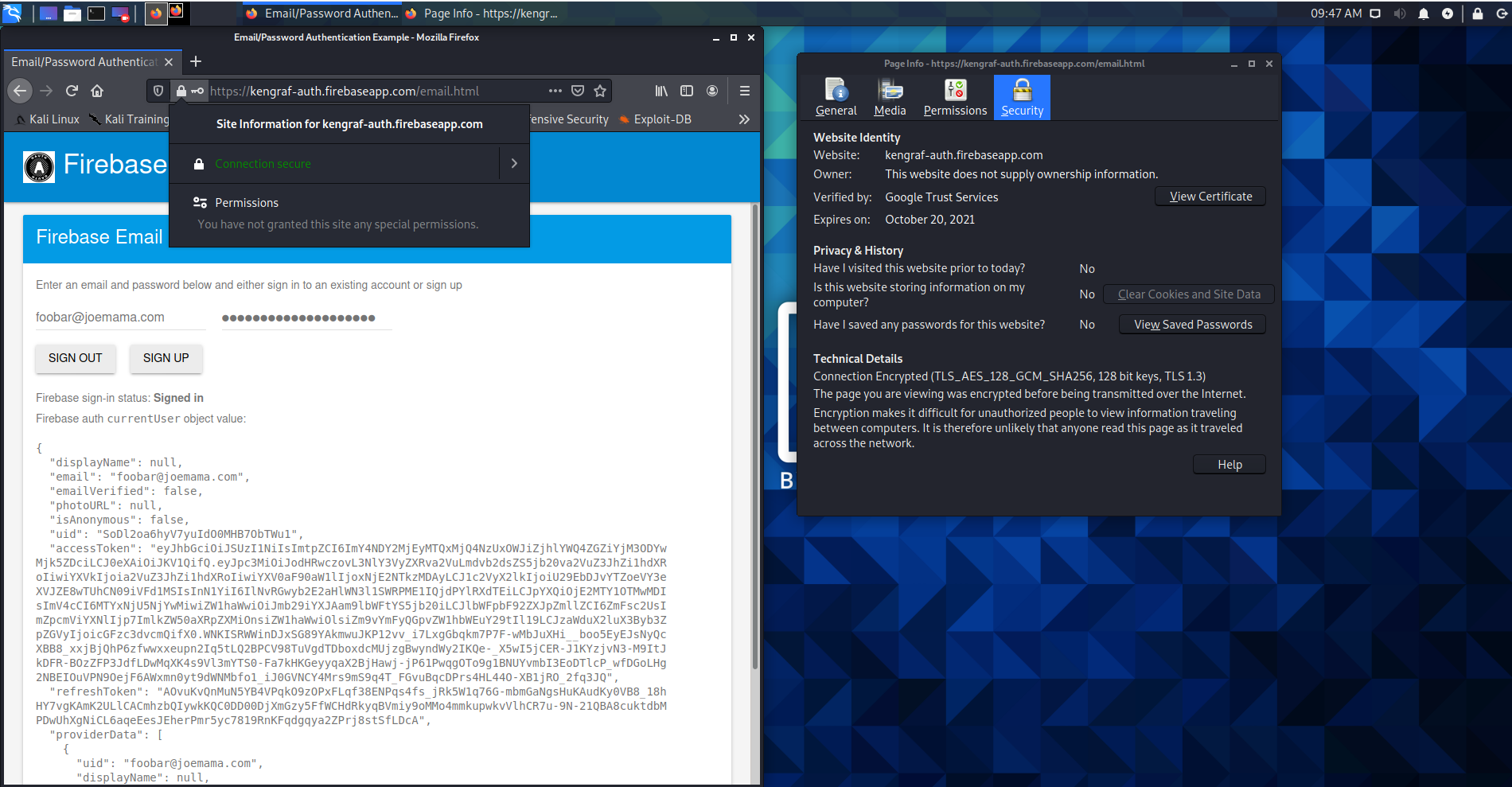
# IT666 - Identity Lab

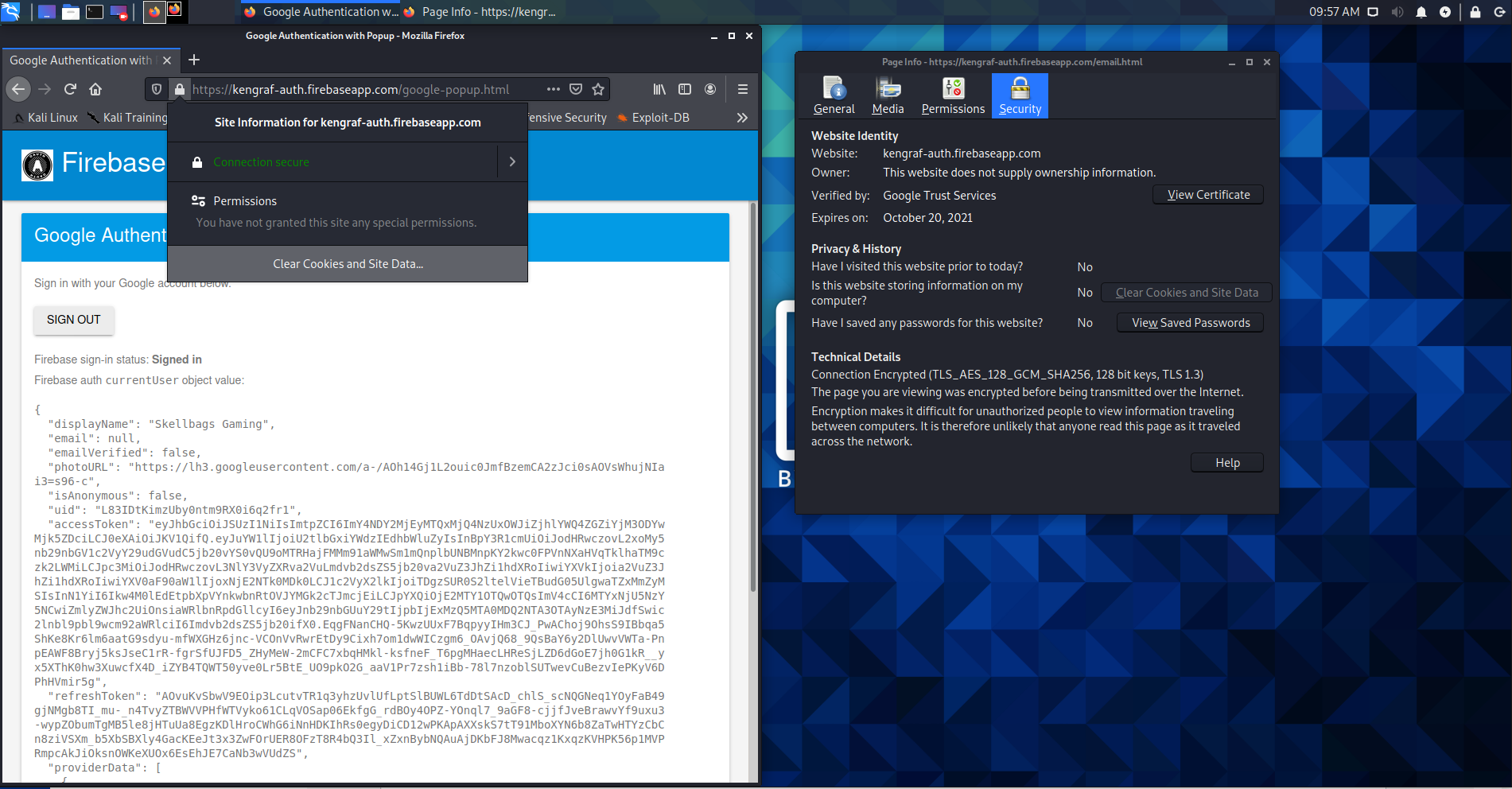
Ryan J. Skelly – [rjs1070@wildcats.unh.edu](mailto:rjs1070@wildcats.unh.edu) – 3/24/2021

**Lab report**:

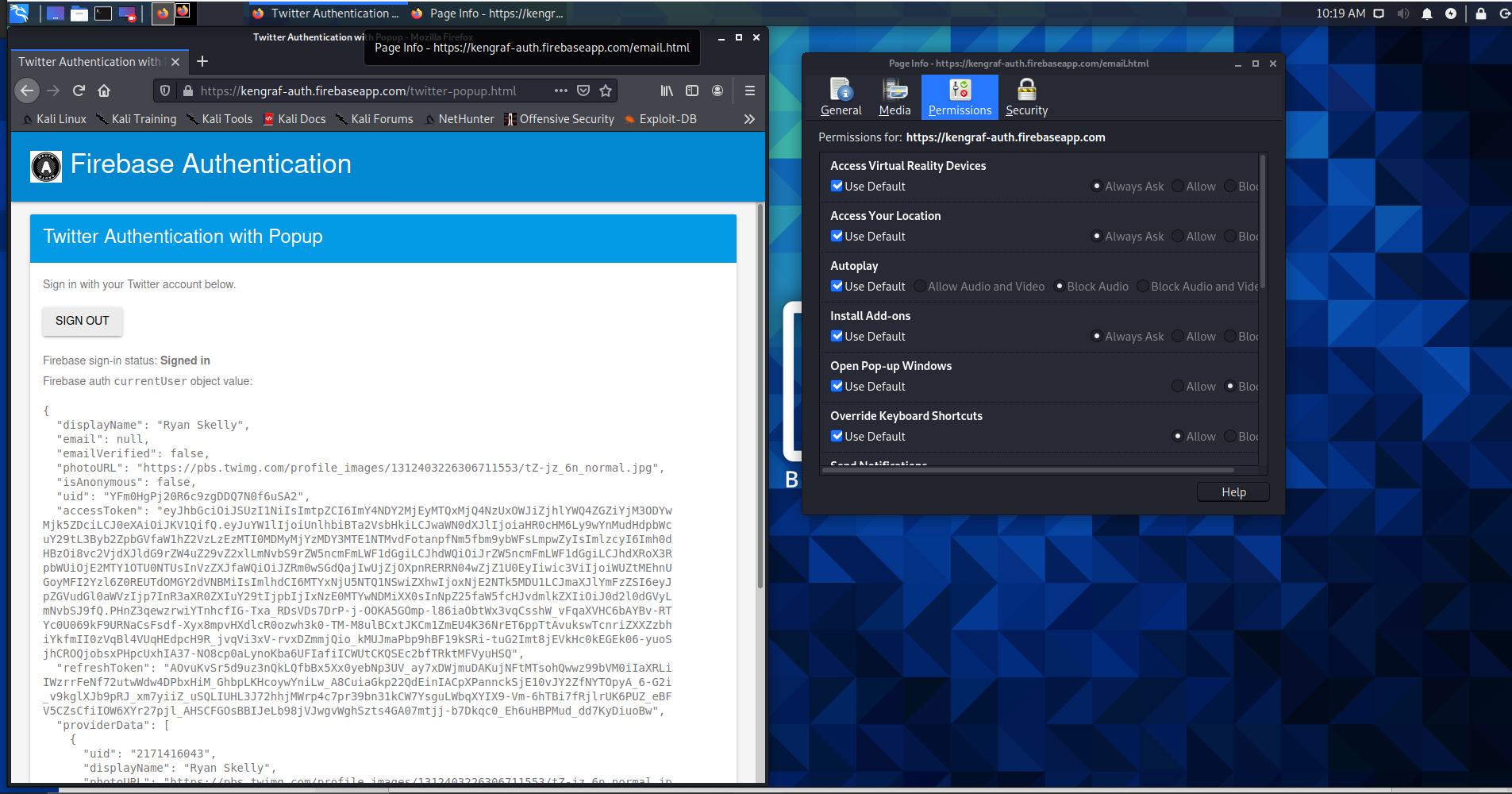
Task 1.1 (10 points) Provide a screen shot of the sign up response.



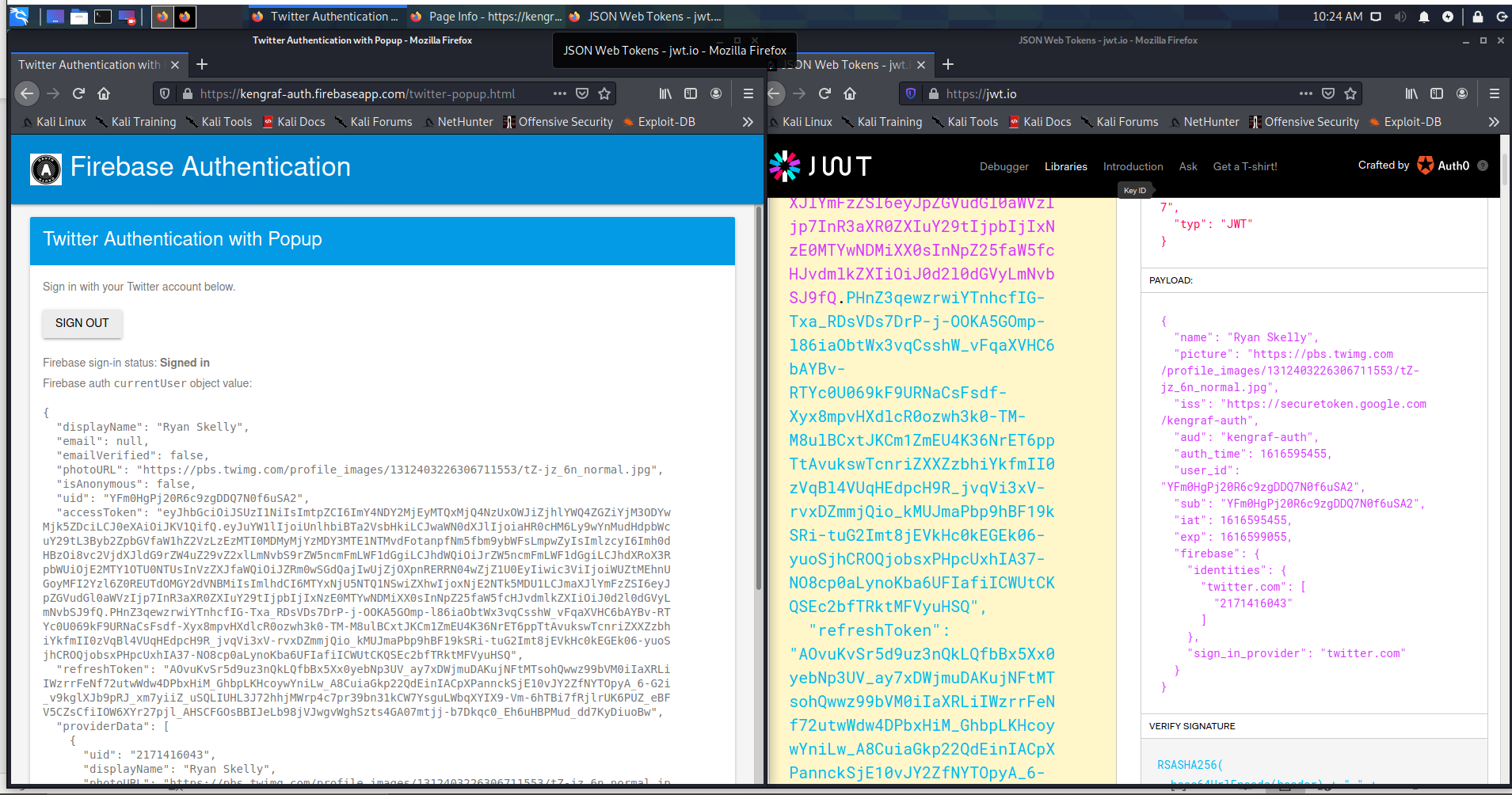
Task 2.1: (5 points) Include a screenshot of the approval page from your identity provider.



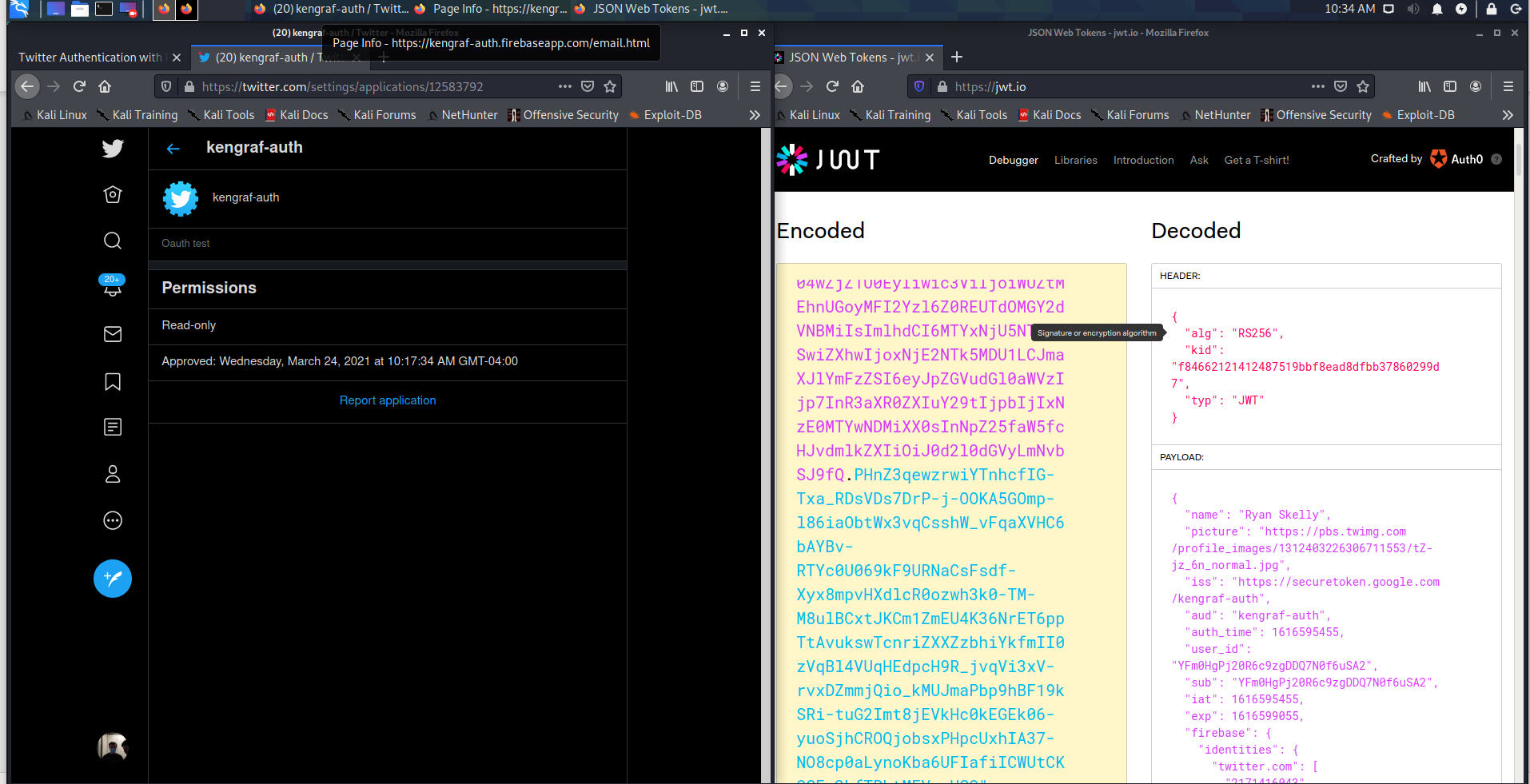
Task 2.2: (5 points) Include a screenshot of successfully logging into Firebase with OpenID.



Task 3.1: (10 points) Take screenshots of a successful login and the output from decoding the accessToken.



Task 4.1: (10 points) Provide a screenshot of the revocation dialog.



Task 5.1: (10 points) Overview of OpenID in your own words

OpenID allows for web apps to divert password management services away from the applications main service, this for the login info to be outsourced to another login service, and the web app acts as the middleman

Task 5.2: (10 points) Description of where your password is stored

My password is stored somewhere on twitter “secure” servers. The web app contacts via API calls to twitter for verification that I exist, and then once I verify myself through their login service, they send data back confirming that I am who I say I am and that it was a valid login

Task 5.3: (10 points) Explain why (or why not) you would recommend OpenID to others.

For web developers just starting out, this would simplify the process of building a secure website, and do it in half to less of the time it would take to set up an even smaller login service, that wouldn’t be as secure, or fast.

Task 5.4: (10 points) Explain how secure you feel using OpenID and what sites you would use it for.

Most larger traffic sites, as well as any email links that I click for password management.

Task 5.5: (10 points) Explain whether or not you would access your banking information with OpenID.

Yes, and credit card information as well, only because I use both Grammarly and Speechify, two extensions that track text input.

**Summary Task**: Answer any two of Tips/Questions in the lab (5 points each).

#1 **Tips/Questions:**Is this step necessary given the “kgraf-auth” app does not do anything? What problem exists if the secret keys for the “kgraf-auth” are compromised?

If secret keys are compromised then hackers would be able to use the keys to pass for you with prefabricated cookies, removing the authorization times out the lifetime of the key, making a cookie claiming to be you would get discarded by the login service.

#2 **Tips/Questions:**Who are you trusting with your credentials? Is revoking access required functionality?

You’re trusting the OAuth certifying body with your login information and accessing it through api calls rather than actually dealing with the data of the password input. Revoking access is smart to implement because if there are any compromises of secret keys then it would allow for the compromises to be recovered from and not fully lose the users password, that could be potentially taken to other sites to check if it works else where.