FEDERATED AUTHENTICATION

Deliverable for this lab: briefly add the following information to the deliverable

- ***1.1: screenshot of the terminal in which you are running the docker-compose command***
- ***1.2: screenshot(s) of the network requests during an authentication and a statement whether or not these requests match the sequence diagrams from the Gitbook***
- ***1.3: decoded JWT token***
- ***1.4: a screenshot of the error you receive after you have tampered with the SAML***
- ***1.5: a screenshot of the error you receive after you have tampered with the JWT***
- ***1.6: a screenshot of the Auth0 configuration for your vanillaJS application***

1.1 Get up and running

Get https://github.com/Mich-b/ModernACREST up and running. You need docker and docker-compose for this.

Since we are using local development certificates, you will be getting some certificate errors – ignore them if possible. (**Only ignore certificate errors if you know why they appear and never provide confidential information on such websites**. We come back on this topic in one of the next courses.).

Credentials to login with OIDC are:

- username:student1
- password:Student

Credentials to login with SAML are:

- username:student1
- password:password

1.2 Understanding OIDC

Browse to http://singlepageapp:8082 , and investigate the network flows using the Developer console or using Burp suite during log-in. Match it to the sequence diagrams that are given in https://apwt.gitbook.io/software-security/access-control-advanced/000introauthenticationadvanced/001federation#openid-connect-oidc . Do you see differences?

1.3 Understanding JWTs

After logging in, you should have received two tokens:

- an identity token
- an access token

Inspect the **access token** via https://jwt.io/ . Do you understand the contents? Do you see how long the access token is valid?

1.4 Tampering with the SAML token

Now it's time to try to tamper with the SAML token.

When authenticating using SAML, there is a SAML token passed from the SAML identity provider (shibbolethidp:8090) to the movieswebapp through the browser. Find the request where this SAML token is passed.

Try to understand what this means: the movieswebapp trusts the fact that some identity provider (the shibbolethidp in our case) has authenticated you and trusts the information that is passed via the token. If you can tamper with this token, you can be whoever you like to be! Try to change this SAML token to contain a different username (e.g. ‘attacker’). Does it work? What error do you receive?

1.5 Tampering with the JWT token

Now do the same with the JWT token that is passed when logging in to http://singlepageapp:8082.

1.6 Deploy your own app and connect it to Auth0

While we attempt to remain vendor-agnostic, Auth0 has some very nice tutorials available that are useful to follow when learning about federated authentication.

Follow their vanillaJS quick start: https://auth0.com/docs/quickstart/spa/vanillajs/01-login. Choose ‘I want to explore a sample app’.