Learning Outcome: Students will gain an experience in implementing their own authentication protocol.

Lab Set Up:

For this lab, you will need four virtual machines. The OS you choose for each doesn't matter, although some will be easier than others.

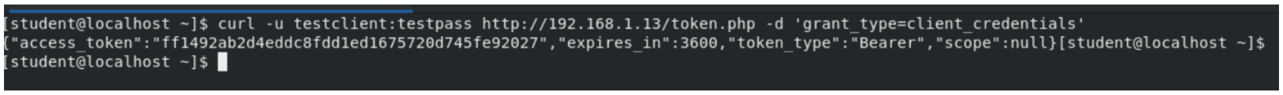
1. A VM that will act as a client. You will have to write a client application that takes user credentials and passes them to the authentication server.
2. A VM that will act as an authentication server. You will have to write a server that accepts connections from the client and then passes credentials to the OAUTH provider. It will encrypt the response as described below.
3. A VM that will act an as an OAUTH provider. This will need to run Apache and MySQL. You do not need to write any code for this VM (unless you want to write a PHP script to enroll new users, but this can also be done by directly adding users to the database by hand).
    a. You are not expected to code your own OAUTH provider. It is recommended you use the tutorial at the following link: https://bshaffer.github.io/oauth2-server-php-docs/
4. A VM that will act as an application server. This will take a valid encrypted OAUTH token and decrypt it using a secret key shared with the authentication server.

O-Kerberos Authentication Protocol
1. The user enters credentials in to the client application.
2. The client application sends credentials to the authentication server.
3. The authentication server uses the credentials to make an HTTP request to the appropriate PHP script on the OAUTH provider.
4. If the credentials are not valid, the OAUTH provider will not return an OAUTH token to the authentication server and the authentication server should return the following JSON to the client, indicating an unsuccessful login: {“auth”:”fail”, “token”:””}.
5. If the credentials are valid, the OAUTH provider will return an OAUTH token in a JSON response to the authentication server.
6. The authentication server should encrypt the JSON response containing the valid OAUTH token with a secret key known to the authentication server and the application server.
7. The authentication server should then construct the following JSON response: {“auth”:”success”, “token”:”<encrypted JSON RESPONSE>”}
8. The authentication server will encrypt this JSON response using the SHA256 hash of the user's password and return it to the client.
9. The client will decrypt the response sent to it from the authentication server and send the encrypted JSON containing the OAUTH token to the application server.
10. The application server will decrypt the encrypted JSON response containing the token using its secret key, indicating that the token came from the authentication server.

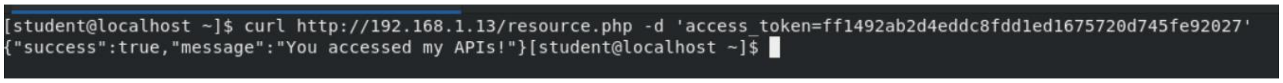
Please provide your steps with detailed screenshots for the following implementations

1. An OAUTH provider was properly deployed on one virtual host.
   I don't know how to really prove the OAUTH provider is deployed properly except providing these. Also, it doesn't say what I should provide.

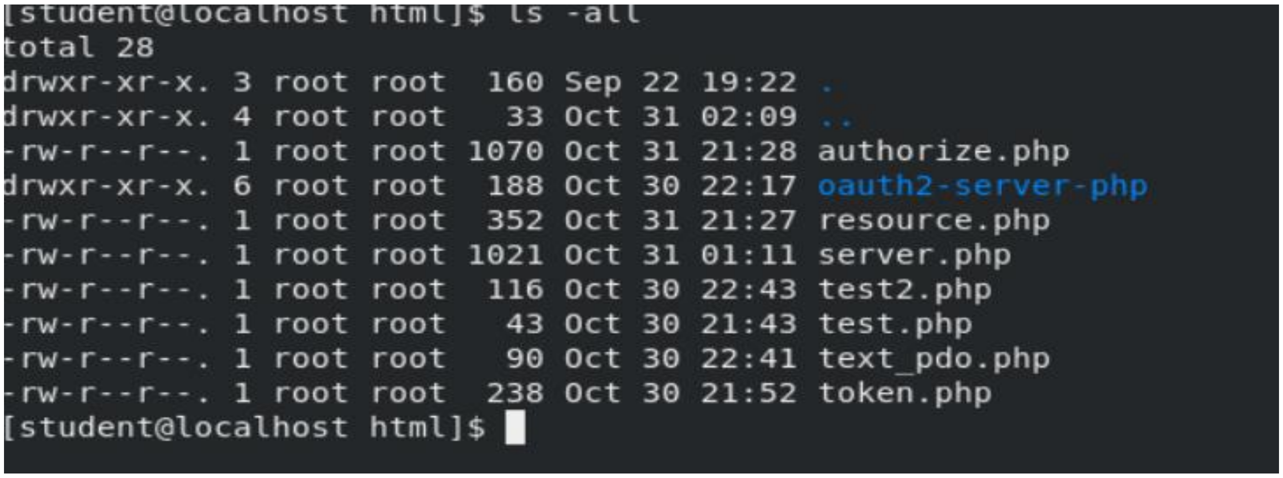
   Here is some connection to the provider to provide tokens and use it. Also, I have attached a picture showing the files that I created for the provider.



```
[student@localhost ~]$ curl -u testclient:testpass http://192.168.1.13/token.php -d 'grant_type=client_credentials'
{"access_token":"ff1492ab2d4eddc8fdd1ed1675720d745fe92027","expires_in":3600,"token_type":"Bearer","scope":null}[student@localhost ~]$
[student@localhost ~]$
```



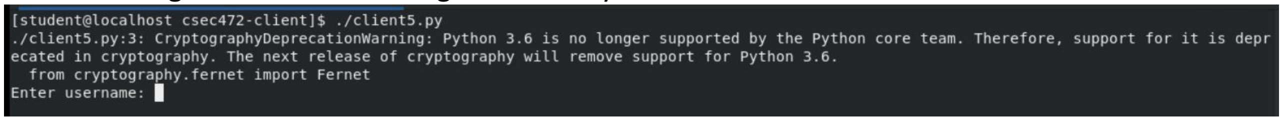
```
[student@localhost ~]$ curl http://192.168.1.13/resource.php -d 'access_token=ff1492ab2d4eddc8fdd1ed1675720d745fe92027'
{"success":true,"message":"You accessed my APIs!"}[student@localhost ~]$
```



```
[student@localhost html]$ ls -all
total 28
drwxr-xr-x. 3 root root  160 Sep 22 19:22 .
drwxr-xr-x. 4 root root   33 Oct 31 02:09 ..
-rw-r--r--. 1 root root 1070 Oct 31 21:28 authorize.php
drwxr-xr-x. 6 root root  188 Oct 30 22:17 oauth2-server-php
-rw-r--r--. 1 root root  352 Oct 31 21:27 resource.php
-rw-r--r--. 1 root root 1021 Oct 31 01:11 server.php
-rw-r--r--. 1 root root  116 Oct 30 22:43 test2.php
-rw-r--r--. 1 root root   43 Oct 30 21:43 test.php
-rw-r--r--. 1 root root   90 Oct 30 22:41 text_pdo.php
-rw-r--r--. 1 root root  238 Oct 30 21:52 token.php
[student@localhost html]$
```

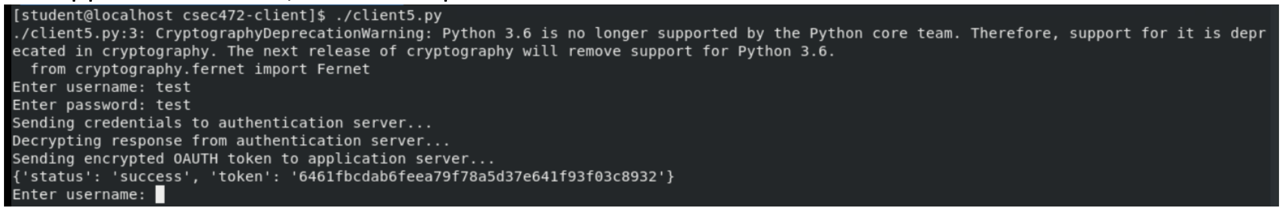
2. The O-Kerberos client behaves as intended.

Here showing the client is working without any issues.



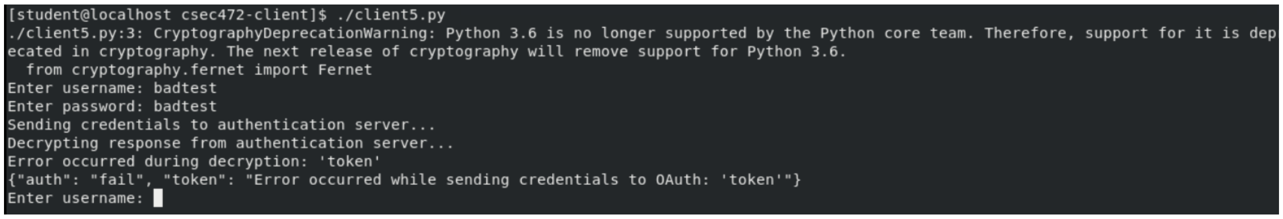
```
[student@localhost csec472-client]$ ./client5.py
./client5.py:3: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is depr
ecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
Enter username: 
```

Here the picture is showing the I successfully connected to the authentication server using valid username and password and it got a response from the authentication server and then send it to the application server, and then I printed the token as confirmation.



```
[student@localhost csec472-client]$ ./client5.py
./client5.py:3: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is depr
ecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
Enter username: test
Enter password: test
Sending credentials to authentication server...
Decrypting response from authentication server...
Sending encrypted OAUTH token to application server...
{'status': 'success', 'token': '6461fbcdab6feea79f78a5d37e641f93f03c8932'}
Enter username: 
```

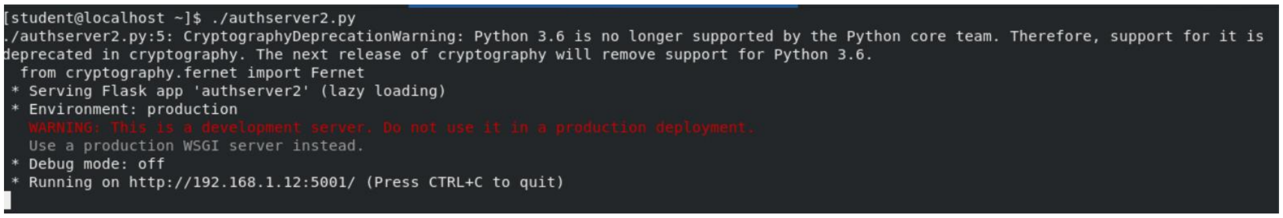
Here the picture shows that I failed the connection because of the invalid password and username.



```
[student@localhost csec472-client]$ ./client5.py
./client5.py:3: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is dep
ecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
Enter username: badtest
Enter password: badtest
Sending credentials to authentication server...
Decrypting response from authentication server...
Error occurred during decryption: 'token'
{"auth": "fail", "token": "Error occurred while sending credentials to OAuth: 'token'"}
Enter username: 
```

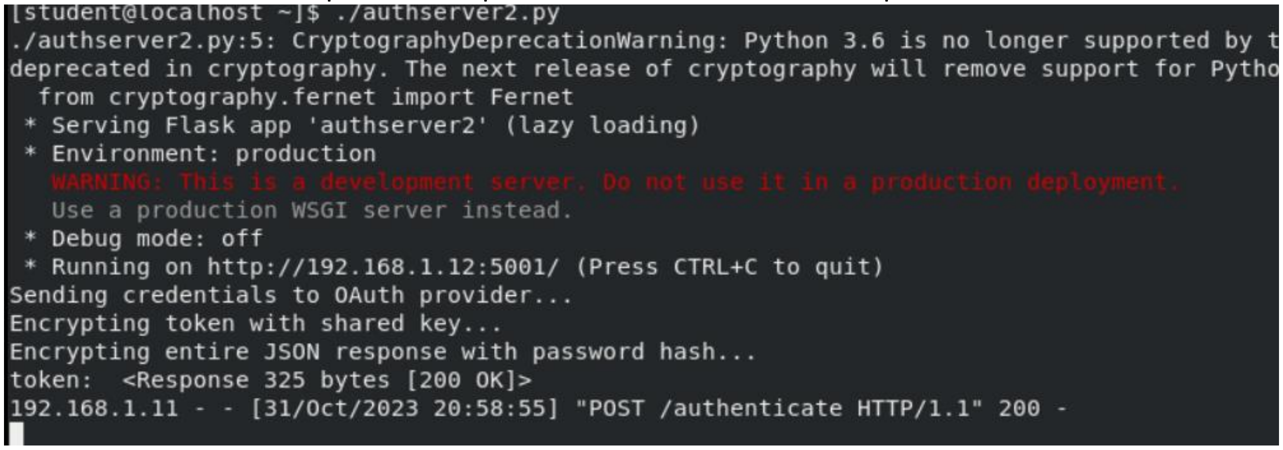
3. The O-Kerberos authentication server behaves as intended.

Here the picture shows that the authentication server is working perfectly.



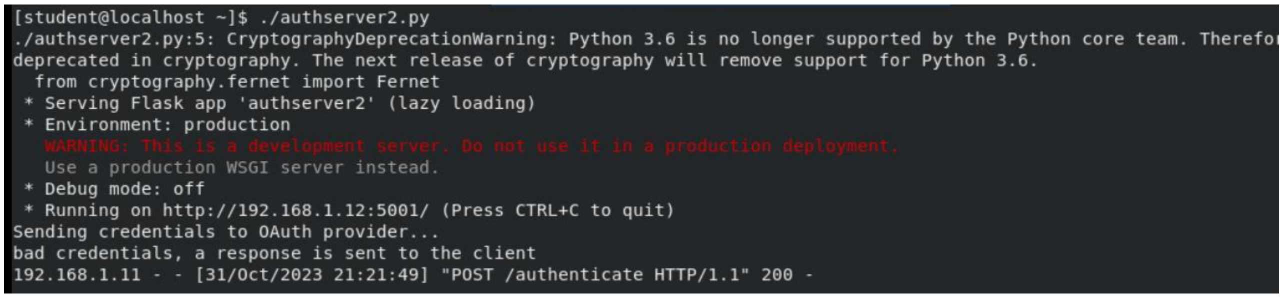
```
[student@localhost ~]$ ./authserver2.py
./authserver2.py:5: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is
deprecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
 * Serving Flask app 'authserver2' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.12:5001/ (Press CTRL+C to quit)
```

Here the picture shows that the authentication server got a valid connection from the client and then sent the made a request to the provider and then send the response back to the client.



```
[student@localhost ~]$ ./authserver2.py
./authserver2.py:5: CryptographyDeprecationWarning: Python 3.6 is no longer supported by t
deprecated in cryptography. The next release of cryptography will remove support for Pytho
  from cryptography.fernet import Fernet
 * Serving Flask app 'authserver2' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.12:5001/ (Press CTRL+C to quit)
Sending credentials to OAuth provider...
Encrypting token with shared key...
Encrypting entire JSON response with password hash...
token:  <Response 325 bytes [200 OK]>
192.168.1.11 - - [31/Oct/2023 20:58:55] "POST /authenticate HTTP/1.1" 200 -
```

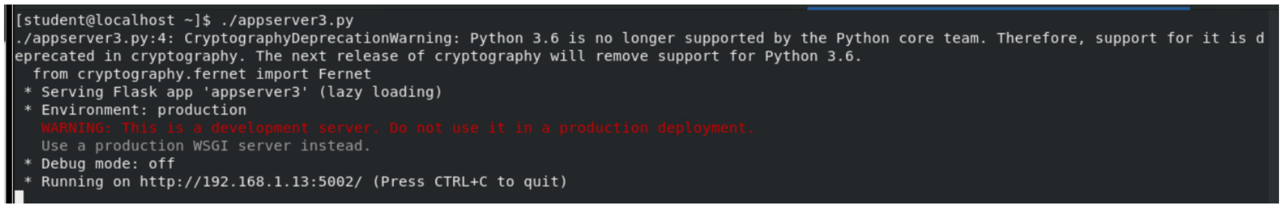
Here the picture shows that the authentication server got a invalid connection from the client, and got a response from the provider stating that it is bad credentials, and then sent it back to the client.



```
[student@localhost ~]$ ./authserver2.py
./authserver2.py:5: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefo
deprecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
 * Serving Flask app 'authserver2' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.12:5001/ (Press CTRL+C to quit)
Sending credentials to OAuth provider...
bad credentials, a response is sent to the client
192.168.1.11 - - [31/Oct/2023 21:21:49] "POST /authenticate HTTP/1.1" 200 -
```

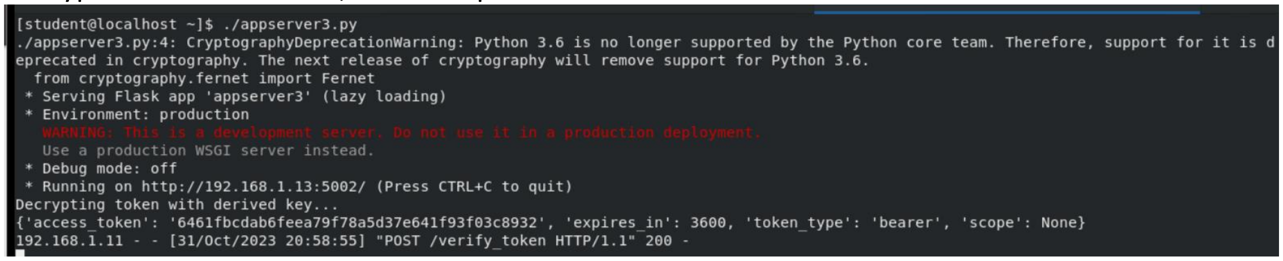
4. The O-Kerberos application server behaves as intended.

Here the picture shows that the application server is working perfectly.



```
[student@localhost ~]$ ./appserver3.py
./appserver3.py:4: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is d
eprecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
 * Serving Flask app 'appserver3' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.13:5002/ (Press CTRL+C to quit)
```

Here the picture shows that the application server got a connection from the client, and then decrypted the token. Also, I made it print the token as confirmation



```
[student@localhost ~]$ ./appserver3.py
./appserver3.py:4: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is d
eprecated in cryptography. The next release of cryptography will remove support for Python 3.6.
  from cryptography.fernet import Fernet
 * Serving Flask app 'appserver3' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.13:5002/ (Press CTRL+C to quit)
Decrypting token with derived key...
{'access_token': '6461fbcdab6feea79f78a5d37e641f93f03c8932', 'expires_in': 3600, 'token_type': 'bearer', 'scope': None}
192.168.1.11 - - [31/Oct/2023 20:58:55] "POST /verify_token HTTP/1.1" 200 -
```

5. Improve the security of the O-Kerberos Protocol by
   1) Creating a public/private key pair for the authentication server and having the client encrypt the credentials sent to the authentication server using the authentication server's public key,
   2) Having the authentication server communicate to the OAUTH provider using HTTPS, and
   3) Having the application server validate the OAUTH token with the OAUTH provider using HTTPS. Note that validation is *not* required unless you're doing the extra credit.