

**Rochester Institute of Technology**



**CSEC 472 Authentication and Security Models**

**BS in Cybersecurity**

**Department of Cybersecurity**

**Lab 2: OAuth**

**Learning Outcome:**

Students will gain experience in implementing an Oauth-type 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 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).
   1. 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 into 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.

Credits: <https://github.com/Sma-Das/O-Kerberos>

# Deliverables:

1. A copy of the sign-off form on the next page.
2. A detailed lab report with screen captures.
3. Workload distribution document.