# Lab 10: Future Crypto

## Details

Aim: To provide a foundation in some of the up-and-coming methods in cryptography.

## A JWT

The JSON Web Token can be used to grant access rights. It is signed using a secret passphrase. Using node.js, create the following program:

var args = process.argv;

var sec='fff';

var pay="{ foo: \'bar\'}";

if (args.length>1) pay=args[2];

if (args.length>2) sec=args[3];

console.log("Message:\t",pay)

console.log("Passphrase:\t",sec)

var jwt = require('jwt-simple');

var payload = pay ;

var secret = sec;

// encode

var token = jwt.encode(payload, secret);

console.log("Token: ",token);

// decode

var decoded = jwt.decode(token, secret);

console.log("Decoded: ",decoded);

Now, match the following parameters to the correct signing key:

napier123

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.InsgZm9vOiAnYmFyJ30i.essZj9zuUdD25nKWxrROQ-M4o7sv532aZ9-xTXANTzo

<https://requests-oauthlib.readthedocs.io/en/latest/examples/real_world_example.html#real-example>

The following is an Oauth 2 method to create a federated ID login

from requests\_oauthlib import OAuth2Session

from flask import Flask, request, redirect, session, url\_for

from flask.json import jsonify

import os

app = Flask(\_\_name\_\_)

# This information is obtained upon registration of a new GitHub OAuth

# application here: https://github.com/settings/applications/new

client\_id = "<your client key>"

client\_secret = "<your client secret>"

authorization\_base\_url = 'https://github.com/login/oauth/authorize'

token\_url = 'https://github.com/login/oauth/access\_token'

@app.route("/")

def demo():

"""Step 1: User Authorization.

Redirect the user/resource owner to the OAuth provider (i.e. Github)

using an URL with a few key OAuth parameters.

"""

github = OAuth2Session(client\_id)

authorization\_url, state = github.authorization\_url(authorization\_base\_url)

# State is used to prevent CSRF, keep this for later.

session['oauth\_state'] = state

return redirect(authorization\_url)

# Step 2: User authorization, this happens on the provider.

@app.route("/callback", methods=["GET"])

def callback():

""" Step 3: Retrieving an access token.

The user has been redirected back from the provider to your registered

callback URL. With this redirection comes an authorization code included

in the redirect URL. We will use that to obtain an access token.

"""

github = OAuth2Session(client\_id, state=session['oauth\_state'])

token = github.fetch\_token(token\_url, client\_secret=client\_secret,

authorization\_response=request.url)

# At this point you can fetch protected resources but lets save

# the token and show how this is done from a persisted token

# in /profile.

session['oauth\_token'] = token

return redirect(url\_for('.profile'))

@app.route("/profile", methods=["GET"])

def profile():

"""Fetching a protected resource using an OAuth 2 token.

"""

github = OAuth2Session(client\_id, token=session['oauth\_token'])

return jsonify(github.get('https://api.github.com/user').json())

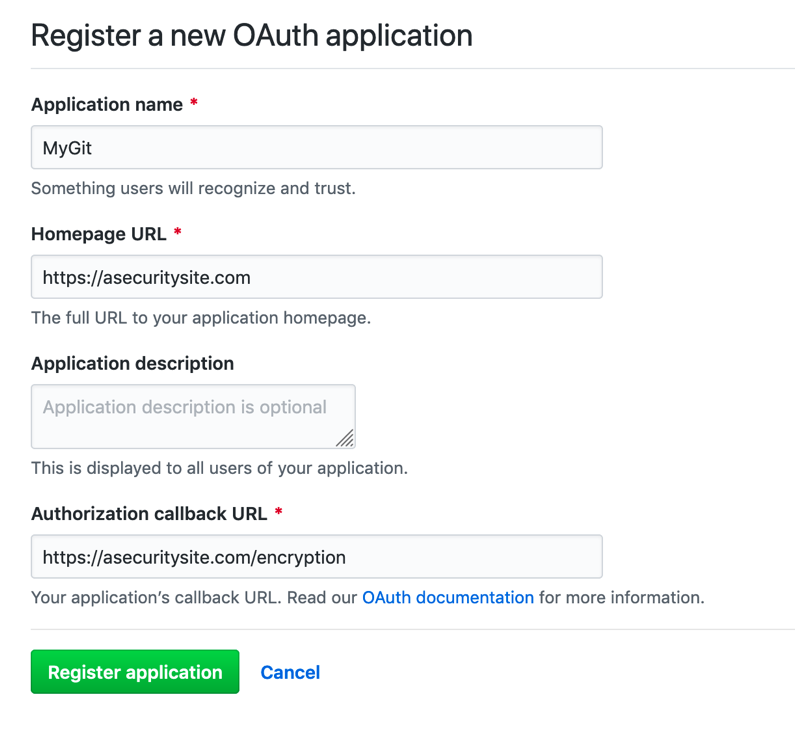
if \_\_name\_\_ == "\_\_main\_\_":

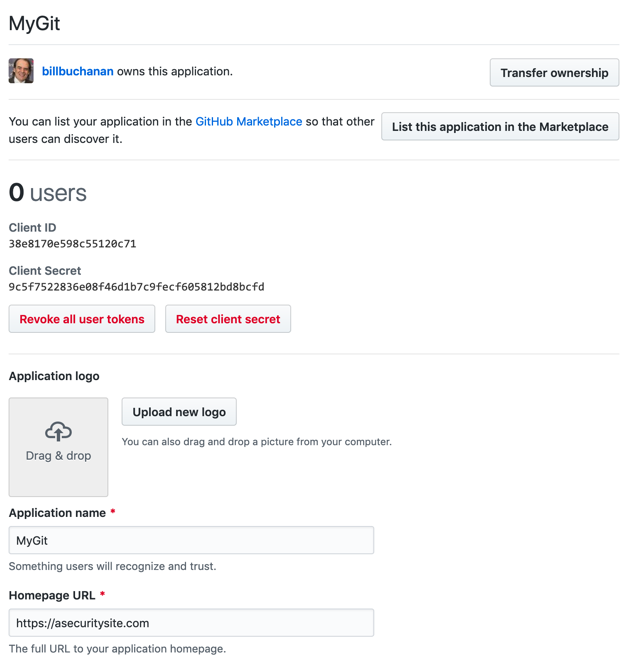
# This allows us to use a plain HTTP callback

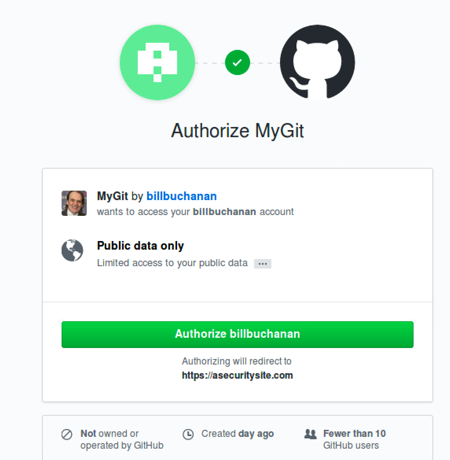
os.environ['OAUTHLIB\_INSECURE\_TRANSPORT'] = "1"

app.secret\_key = os.urandom(24)

app.run(debug=True)







Now clear the cache in Firefox.

## Light-weight crypto

**L2.1** NIST has defined a number of possible contenders for light-weight crypto methods. By searching the Internet, can you find those that have been recommended:

* sudo apt update

Next, install a few prerequisite packages which let apt use packages over HTTPS:

* sudo apt install apt-transport-https ca-certificates curl software-properties-common

Then add the GPG key for the official Docker repository to your system:

* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Add the Docker repository to APT sources:

* sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

Next, update the package database with the Docker packages from the newly added repo:

* sudo apt update

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

* apt-cache policy docker-ce

You'll see output like this, although the version number for Docker may be different:

Output of apt-cache policy docker-ce

docker-ce:

Installed: (none)

Candidate: 18.03.1~ce~3-0~ubuntu

Version table:

18.03.1~ce~3-0~ubuntu 500

500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages

Notice that docker-ce is not installed, but the candidate for installation is from the Docker repository for Ubuntu 18.04 (bionic).

Finally, install Docker:

* sudo apt install docker-ce

mkdir docker\_sshd

nano Docker

FROM ubuntu:16.04

RUN apt-get update && apt-get install -y openssh-server

RUN mkdir /var/run/sshd

RUN echo 'root:THEPASSWORDYOUCREATED' | chpasswd

RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config

# SSH login fix. Otherwise user is kicked off after login

RUN sed 's@session\s\*required\s\*pam\_loginuid.so@session optional pam\_loginuid.so@g' -i /etc/pam.d/sshd

ENV NOTVISIBLE "in users profile"

RUN echo "export VISIBLE=now" >> /etc/profile

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]

docker build -t my\_sshd .

Step 10/10 : CMD ["/usr/sbin/sshd", "-D"]

---> Running in 982e91d7a665

Removing intermediate container 982e91d7a665

---> e90cde8a95d3

Successfully built e90cde8a95d3

Successfully tagged my\_sshd:latest

napier@napier-virtual-machine:~/Docker\_sshd$ docker run -d -P --name test\_sshd my\_sshd

2e1ee60deb3f44f2c4a6af5ebd0e32f9882ba4e03042f0eb30285f74e49ced39

napier@napier-virtual-machine:~/Docker\_sshd$ docker port test\_sshd 220.0.0.0:32769

napier@napier-virtual-machine:~/Docker\_sshd$ ssh root@localhost -p 32769

The authenticity of host '[localhost]:32769 ([127.0.0.1]:32769)' can't be established.

ECDSA key fingerprint is SHA256:WvbGw6gNNJzLksG87PWQ3yg+YvHBQc9PLFJl3wPOWnM.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[localhost]:32769' (ECDSA) to the list of known hosts.

root@localhost's password:

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-142-generic x86\_64)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by

applicable law.

root@2e1ee60deb3f:~# ls

root@2e1ee60deb3f:~# mkdir test

root@2e1ee60deb3f:~#

napier@napier-virtual-machine:~/Docker\_sshd$ docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE

my\_sshd latest e90cde8a95d3 4 minutes ago 196MB

friendlyhello latest 290382d66d7b 19 minutes ago 131MB

python 2.7-slim 48e3247f2a19 5 days ago 120MB

ethereum/solc stable 1b65904c442c 2 weeks ago 6.48MB

hello-world latest fce289e99eb9 3 months ago 1.84kB

ubuntu 16.04 b0ef3016420a 3 months ago

napier@napier-virtual-machine:~/Docker\_sshd$ docker container stop test\_sshd

test\_sshd

napier@napier-virtual-machine:~/Docker\_sshd$ docker container rm test\_sshd

test\_sshd

napier@napier-virtual-machine:~/Docker\_sshd$ docker image rm eg\_sshd

Error: No such image: eg\_sshd

napier@napier-virtual-machine:~/Docker\_sshd$ docker image rm my\_sshd

Untagged: my\_sshd:latest

Deleted: sha256:e90cde8a95d3bff3b042b478adcceb92176a3e629499def5e8da322774c2cf47

Deleted: sha256:e5a6b2da57c6b7b6c772c305e16b34336de88f12b19b27f16cc7c660f7b7b1ea

Deleted: sha256:0c37ba9e6a6a5295d7549707c8da0c5702c5a579a57b610ad5aeb52259843e13

Deleted: sha256:4e086bc6747636c802f21db9258e6c8f591976c74258a39cb0a310f2f0d6b372

Deleted: sha256:6522e60d28b0b28b92f3e0b8a02c9c55ed307977fc8151af95422d01d34849f4

Deleted: sha256:d98ea9ac1c048d62693d18b7f26497a61f0cd00819f6bf37372dbcdb5bf22646

Deleted: sha256:86cea5b1a64554eb6b77d875461bdd3a09f95434d31ba5ef7e29d0ec94d8728a

Deleted: sha256:a2725f828ea91508991c362737befdb3f095499e4d2c8563eb72ea5bc957447f

Deleted: sha256:0394b2e6286dda3afd3f0d96064158c31ef5be3e2382e62ec038dd1774479d5b

Deleted: sha256:c64692f01b21abee36a9c3708c19578d5e1ec122149ffbec65e976343eabc068

Deleted: sha256:f5b00bcab9a5ccb97815164e7c036defaf08619da84b48dc58bb5ee69ac46446

Deleted: sha256:e0e2413bf9e0a47cfd2057c7abb7c8267a328680a974ffc7cc224691747c9409

Deleted: sha256:2ffb8773dcf1c3167cd7fc8b573d4e6792686b6bd8f0dd5f181d3a41c1fbad01

Deleted: sha256:2ed2f8f1ff06a62f5ef0ac89277db7fc0cdf038390b5d26d4adb99201996a16c

Deleted: sha256:cfa27e3715938d208761829ccc20dda867567b4a4ac06938f8e1d79c32e79b09

|  |  |
| --- | --- |
| token | ".eyJhY2Nlc3MiOltdLCJhdWQiOiJyZWdpc3RyeS5kb2NrZXIuaW8iLCJleHAiOjE1NTQyNDEwMjMsImlhdCI6MTU1NDI0MDcyMywiaXNzIjoiYXV0aC5kb2NrZXIuaW8iLCJqdGkiOiJzNGdSWjdROHdoRUdCalhvQnE4ZSIsIm5iZiI6MTU1NDI0MDQyMywic3ViIjoiIn0.nJ4xci3UqfRoBMnfMXF9tAAvcc8yhpWXDe1DxYKc8Ptr1yJUDmrl3A0ju\_xj9CJyIREaXO600b-4IWJnbs1La4heVcYRXonvAweZY8rkImh50gzk0iOPf66SNV2uff1kg60yTHCB3jUr-gXkiWK6qL8VK6Z9TOo\_hVj6TE0d2owov3lZuxoqjGeXPq7fvkz1k4q2w25QzYEuDdguqLM\_tNPYfopXwEhx3zhbP63-FvdupykTKUY-mwPbhsziieML0kqnAgisk3YiTY2W59zlRNqlSErbrASDePsOOGFfLOoO4UAHErCOFAtyEBlOnJw-HqDRumpazPli8HqCBcyZSQ" |
| access\_token | ".eyJhY2Nlc3MiOltdLCJhdWQiOiJyZWdpc3RyeS5kb2NrZXIuaW8iLCJleHAiOjE1NTQyNDEwMjMsImlhdCI6MTU1NDI0MDcyMywiaXNzIjoiYXV0aC5kb2NrZXIuaW8iLCJqdGkiOiJzNGdSWjdROHdoRUdCalhvQnE4ZSIsIm5iZiI6MTU1NDI0MDQyMywic3ViIjoiIn0.nJ4xci3UqfRoBMnfMXF9tAAvcc8yhpWXDe1DxYKc8Ptr1yJUDmrl3A0ju\_xj9CJyIREaXO600b-4IWJnbs1La4heVcYRXonvAweZY8rkImh50gzk0iOPf66SNV2uff1kg60yTHCB3jUr-gXkiWK6qL8VK6Z9TOo\_hVj6TE0d2owov3lZuxoqjGeXPq7fvkz1k4q2w25QzYEuDdguqLM\_tNPYfopXwEhx3zhbP63-FvdupykTKUY-mwPbhsziieML0kqnAgisk3YiTY2W59zlRNqlSErbrASDePsOOGFfLOoO4UAHErCOFAtyEBlOnJw-HqDRumpazPli8HqCBcyZSQ" |
| expires\_in | 300 |
| issued\_at | "2019-04-02T21:32:03.212881371Z" |

https://www.openpolicyagent.org/docs/http-api-authorization.html

version: '2'

services:

opa:

image: openpolicyagent/opa:0.10.5

ports:

- 8181:8181

# WARNING: OPA is NOT running with an authorization policy configured. This

# means that clients can read and write policies in OPA. If you are

# deploying OPA in an insecure environment, be sure to configure

# authentication and authorization on the daemon. See the Security page for

# details: https://www.openpolicyagent.org/docs/security.html.

command:

- "run"

- "--server"

- "--log-level=debug"

api\_server:

image: openpolicyagent/demo-restful-api:0.2

ports:

- 5000:5000

environment:

- OPA\_ADDR=http://opa:8181

- POLICY\_PATH=/v1/data/httpapi/authz

Then run docker-compose to pull and run the containers.

docker-compose -f docker-compose.yml up

napier@napier-virtual-machine:~$ curl -X PUT --data-binary @example.rego localhost:8181/v1/policies/example

napier@napier-virtual-machine:~$ $url --user alice:password localhost:5000/finance/salary/alice

Success: user alice is authorized

napier@napier-virtual-machine:~$ curl --user bob:password localhost:5000/finance/salary/alice

Success: user bob is authorized

napier@napier-virtual-machine:~$ curl --user bob:password localhost:5000/finance/salary/charlie

Error: user bob is not authorized to GET url /finance/salary/Charlie

napier@napier-virtual-machine:~$ cat example.rego

package httpapi.authz

# bob is alice's manager, and betty is charlie's.

subordinates = {"alice": [], "charlie": [], "bob": ["alice"], "betty": ["charlie"]}

# HTTP API request

import input

default allow = false

# Allow users to get their own salaries.

allow {

input.method = "GET"

input.path = ["finance", "salary", username]

input.user = username

}

# Allow managers to get their subordinates' salaries.

allow {

input.method = "GET"

input.path = ["finance", "salary", username]

subordinates[input.user][\_] = username