Software-Defined Networking

Lab 7

SDN Security and REST-Python

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

Lab Summary

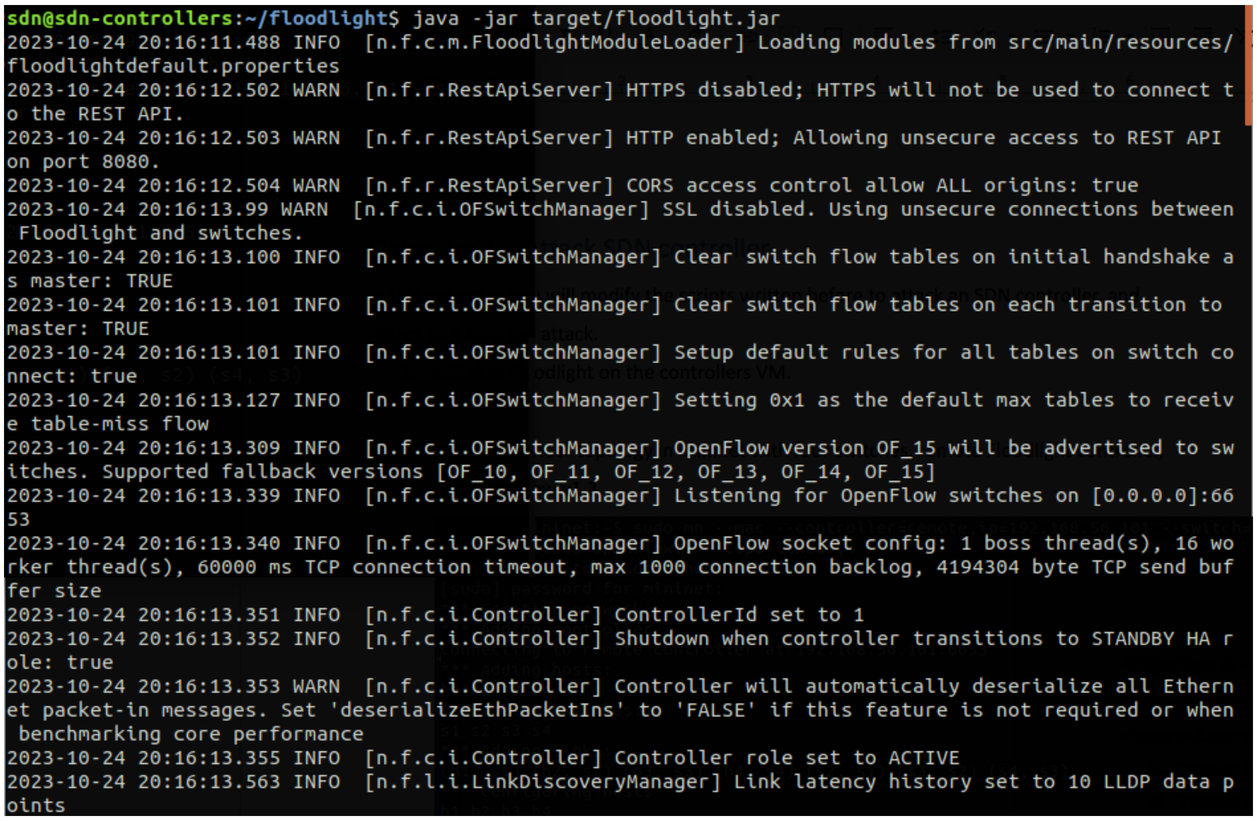
This lab is intended to be an overview of basic TLS and firewall security in SDN. Securing the SDN controller is critical to the security of the entire SDN. TLS is the standard way to secure the southbound communication from the networking devices and the SDN controller. This lab will provide the fundamentals of implementing TLS for OpenFlow communications.

A firewall is a very critical application for any network. It acts as the first/last line of defense against any unauthorized user trying to access or exploit the network. Such users can cause much harm to the networks by adding/changing flow entries to cause misconfigurations, execute DDoS attacks or just silently sniff critical information of the network. The purpose of this lab is to implement a script on Floodlight controller using a python code and understand how rules are implemented to execute certain actions on the packets which match them. The experience gained from completing this lab should be used as a foundation to understanding higher layer firewalls and how software can control the network. In the future, this basic firewall can be enhanced with additional functionality.

Objective 1 – Attack SDN controller

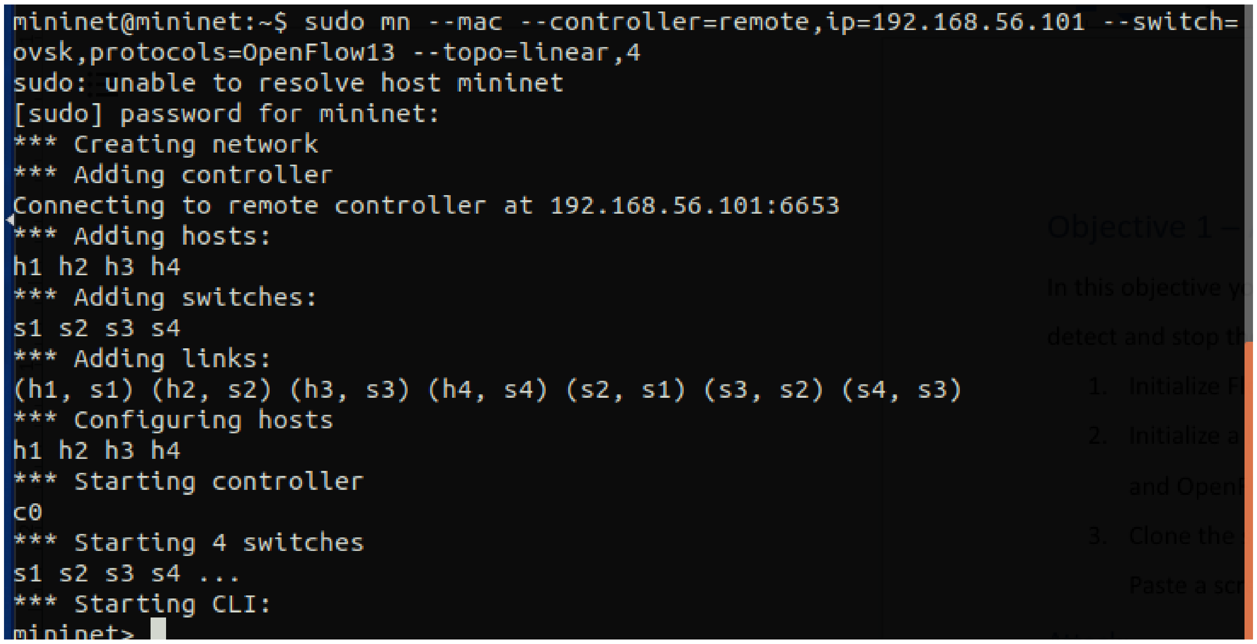
In this objective you will modify the scripts written before to attack an SDN controller, and detect and stop the attack.

1. Initialize Floodlight on the controllers VM.



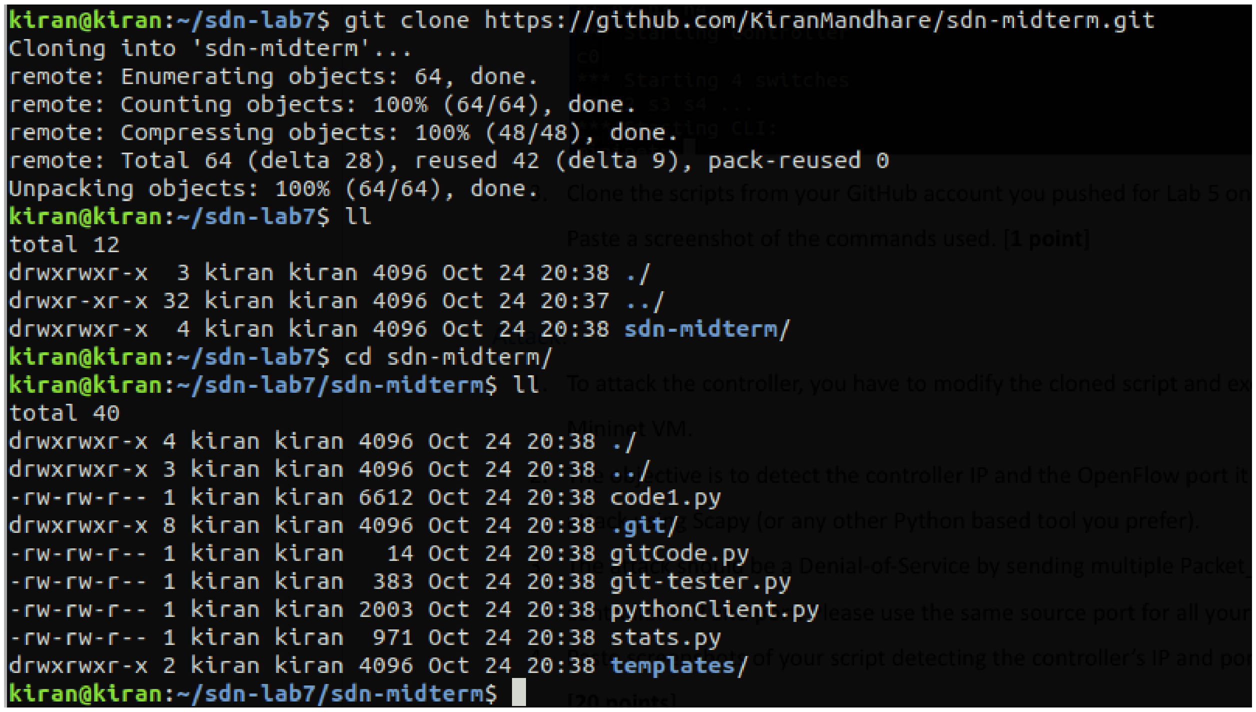
```
sdn@sdn-controllers:~/floodlight$ java -jar target/floodlight.jar
2023-10-24 20:16:11.488 INFO  [n.f.c.m.FloodlightModuleLoader] Loading modules from src/main/resources/
floodlightdefault.properties
2023-10-24 20:16:12.502 WARN  [n.f.r.RestApiServer] HTTPS disabled; HTTPS will not be used to connect t
o the REST API.
2023-10-24 20:16:12.503 WARN  [n.f.r.RestApiServer] HTTP enabled; Allowing unsecure access to REST API
on port 8080.
2023-10-24 20:16:12.504 WARN  [n.f.r.RestApiServer] CORS access control allow ALL origins: true
2023-10-24 20:16:13.99 WARN  [n.f.c.i.OFSwitchManager] SSL disabled. Using unsecure connections between
 Floodlight and switches.
2023-10-24 20:16:13.100 INFO  [n.f.c.i.OFSwitchManager] Clear switch flow tables on initial handshake a
s master: TRUE
2023-10-24 20:16:13.101 INFO  [n.f.c.i.OFSwitchManager] Clear switch flow tables on each transition to
master: TRUE
2023-10-24 20:16:13.101 INFO  [n.f.c.i.OFSwitchManager] Setup default rules for all tables on switch co
nnect: true
2023-10-24 20:16:13.127 INFO  [n.f.c.i.OFSwitchManager] Setting 0x1 as the default max tables to receiv
e table-miss flow
2023-10-24 20:16:13.309 INFO  [n.f.c.i.OFSwitchManager] OpenFlow version OF_15 will be advertised to sw
itches. Supported fallback versions [OF_10, OF_11, OF_12, OF_13, OF_14, OF_15]
2023-10-24 20:16:13.339 INFO  [n.f.c.i.OFSwitchManager] Listening for OpenFlow switches on [0.0.0.0]:66
53
2023-10-24 20:16:13.340 INFO  [n.f.c.i.OFSwitchManager] OpenFlow socket config: 1 boss thread(s), 16 wo
rker thread(s), 60000 ms TCP connection timeout, max 1000 connection backlog, 4194304 byte TCP send buf
fer size
2023-10-24 20:16:13.351 INFO  [n.f.c.i.Controller] ControllerId set to 1
2023-10-24 20:16:13.352 INFO  [n.f.c.i.Controller] Shutdown when controller transitions to STANDBY HA r
ole: true
2023-10-24 20:16:13.353 WARN  [n.f.c.i.Controller] Controller will automatically deserialize all Ethern
et packet-in messages. Set 'deserializeEthPacketIns' to 'FALSE' if this feature is not required or when
 benchmarking core performance
2023-10-24 20:16:13.355 INFO  [n.f.c.i.Controller] Controller role set to ACTIVE
2023-10-24 20:16:13.563 INFO  [n.f.l.i.LinkDiscoveryManager] Link latency history set to 10 LLDP data p
oints
```

2. Initialize a linear topology in Mininet with four switches, remote Floodlight controller, and OpenFlow v1.3.



```
mininet@mininet:~$ sudo mn --mac --controller=remote,ip=192.168.56.101 --switch=
ovsk,protocols=OpenFlow13 --topo=linear,4
sudo: unable to resolve host mininet
[sudo] password for mininet:
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.56.101:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>
```

3. Clone the scripts from your GitHub account you pushed for Lab 5 on the Mininet VM. Paste a screenshot of the commands used. [**1 point**]

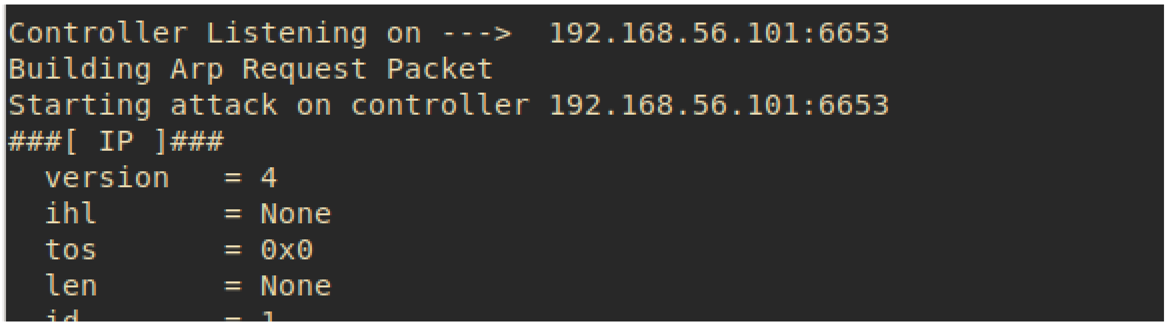


```
kiran@kiran:~/sdn-lab7$ git clone https://github.com/KiranMandhare/sdn-midterm.git
Cloning into 'sdn-midterm'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 64 (delta 28), reused 42 (delta 9), pack-reused 0
Unpacking objects: 100% (64/64), done.
kiran@kiran:~/sdn-lab7$ ll
total 12
drwxrwxr-x  3 kiran kiran 4096 Oct 24 20:38 ./
drwxr-xr-x 32 kiran kiran 4096 Oct 24 20:37 ../
drwxrwxr-x  4 kiran kiran 4096 Oct 24 20:38 sdn-midterm/
kiran@kiran:~/sdn-lab7$ cd sdn-midterm/
kiran@kiran:~/sdn-lab7/sdn-midterm$ ll
total 40
drwxrwxr-x 4 kiran kiran 4096 Oct 24 20:38 ./
drwxrwxr-x 3 kiran kiran 4096 Oct 24 20:38 ../
-rw-rw-r-- 1 kiran kiran 6612 Oct 24 20:38 code1.py
drwxrwxr-x 8 kiran kiran 4096 Oct 24 20:38 .git/
-rw-rw-r-- 1 kiran kiran   14 Oct 24 20:38 gitCode.py
-rw-rw-r-- 1 kiran kiran  383 Oct 24 20:38 git-tester.py
-rw-rw-r-- 1 kiran kiran 2003 Oct 24 20:38 pythonClient.py
-rw-rw-r-- 1 kiran kiran  971 Oct 24 20:38 stats.py
drwxrwxr-x 2 kiran kiran 4096 Oct 24 20:38 templates/
kiran@kiran:~/sdn-lab7/sdn-midterm$
```

Attack:

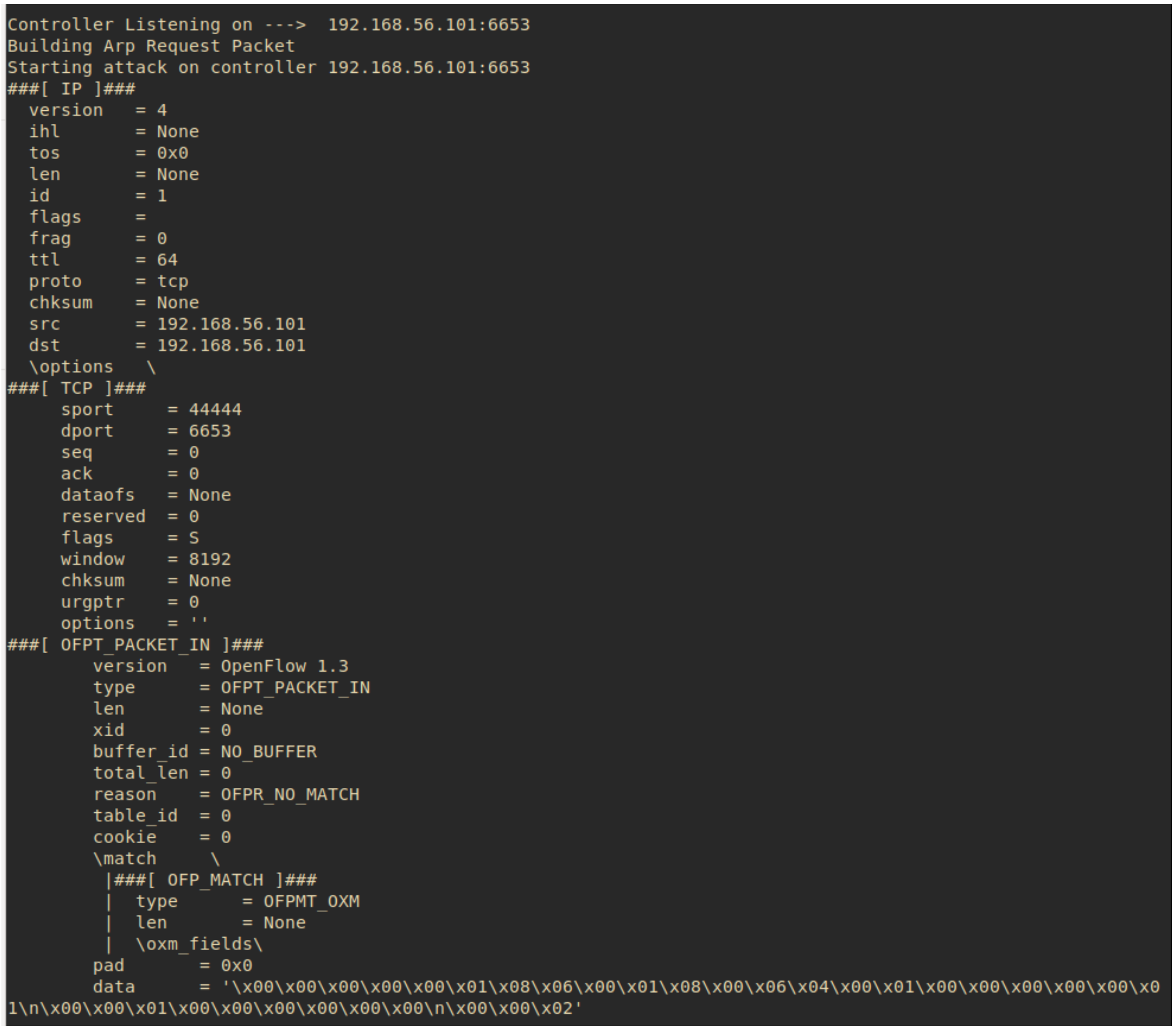
1. To attack the controller, you have to modify the cloned script and execute it on the Mininet VM.
2. The objective is to detects the controller IP and the OpenFlow port it uses and initiate an attack using Scapy (or any other Python based tool you prefer).
3. The attack should be a Denial-of-Service by sending multiple Packet_In messages to the controller's IP and port. Please use the same source port for all your attack packets.
4. Paste screenshots of your script detecting the controller's IP and port, and of the attack. [**20 points**]

   Output attacker.py



```
Controller Listening on --->  192.168.56.101:6653
Building Arp Request Packet
Starting attack on controller 192.168.56.101:6653
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
```

Packet sent for attack:

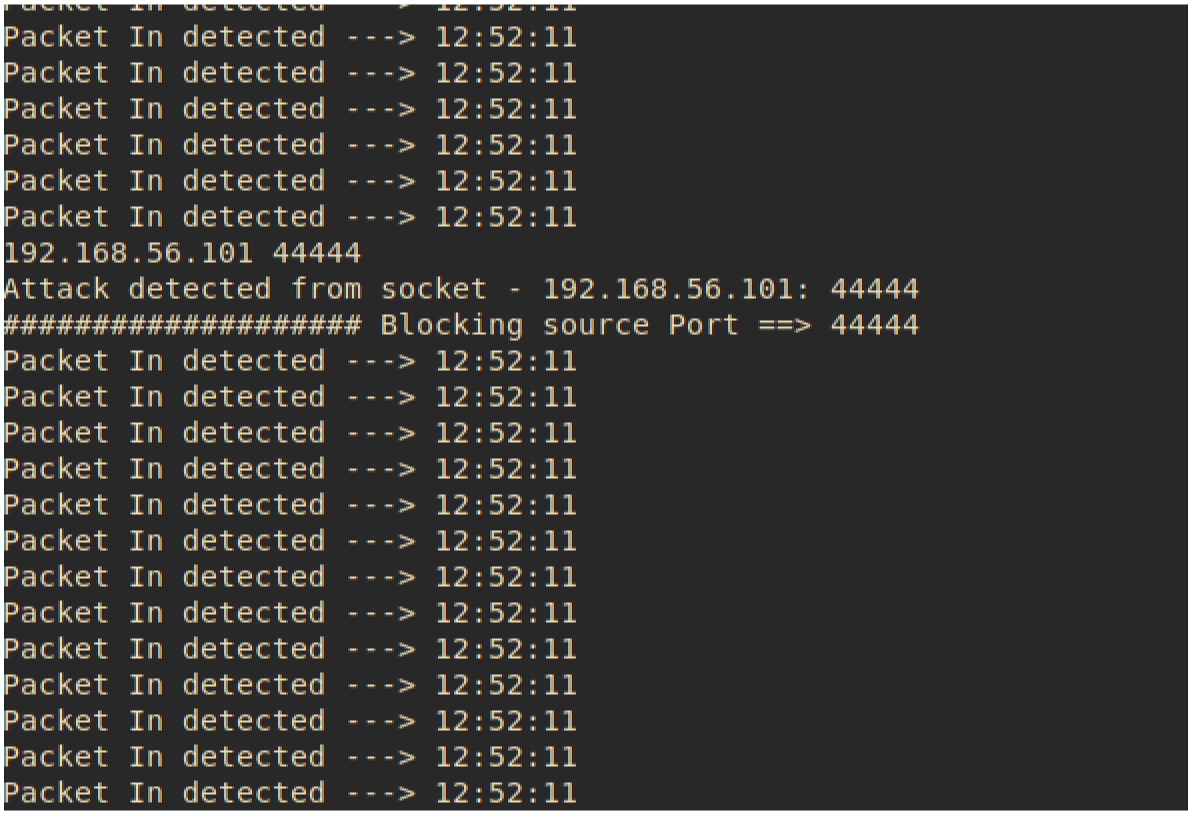


```
Controller Listening on --->  192.168.56.101:6653
Building Arp Request Packet
Starting attack on controller 192.168.56.101:6653
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 192.168.56.101
  dst       = 192.168.56.101
  \options   \
###[ TCP ]###
     sport     = 44444
     dport     = 6653
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = ''
###[ OFPT_PACKET_IN ]###
        version   = OpenFlow 1.3
        type      = OFPT_PACKET_IN
        len       = None
        xid       = 0
        buffer_id = NO_BUFFER
        total_len = 0
        reason    = OFPR_NO_MATCH
        table_id  = 0
        cookie    = 0
        \match     \
         |###[ OFP_MATCH ]###
         |  type      = OFPMT_OXM
         |  len       = None
         |  \oxm_fields\
        pad       = 0x0
        data      = '\x00\x00\x00\x00\x00\x01\x08\x06\x00\x01\x08\x00\x06\x04\x00\x01\x00\x00\x00\x00\x00\x0
1\n\x00\x00\x01\x00\x00\x00\x00\x00\x00\n\x00\x00\x02'
```

Submitted attacker.py

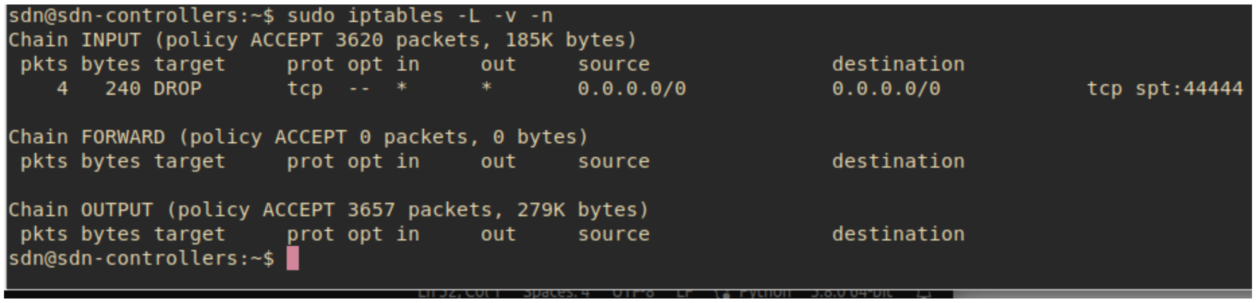
Detect and stop:

1. To detect the attack, use the script from Lab 5 to count the number of Packet_In messages received from a switch IP:port.
2. You must execute the script on the controllers VM and display a message when a threshold (say more than 100 Packet_In messages from one switch IP:port) are detected. Paste screenshots of your script detecting an attack to the controller. [**20 points**]



```
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
192.168.56.101 44444
Attack detected from socket - 192.168.56.101: 44444
#################### Blocking source Port ==> 44444
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
Packet In detected ---> 12:52:11
```

3. To stop the attack, your script should add an iptables on the controllers VM to block packets to the controller's IP and port from the source port of the attack packets. Paste screenshots of the iptables rule added and confirm that the attack has been stopped. [**20 points**]



```
sdn@sdn-controllers:~$ sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 3620 packets, 185K bytes)
 pkts bytes target     prot opt in     out     source               destination
    4   240 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp spt:44444

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 3657 packets, 279K bytes)
 pkts bytes target     prot opt in     out     source               destination
sdn@sdn-controllers:~$ 
```

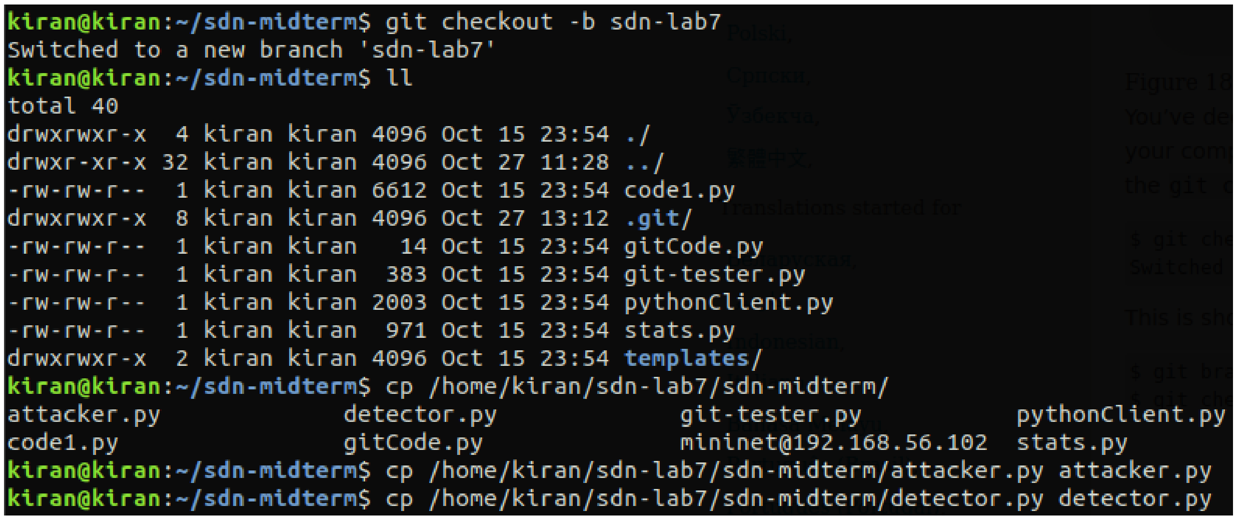
The drops indicate the packets never reach the controller software and are dropped at the host level itself.

submitted detector.py in attachment

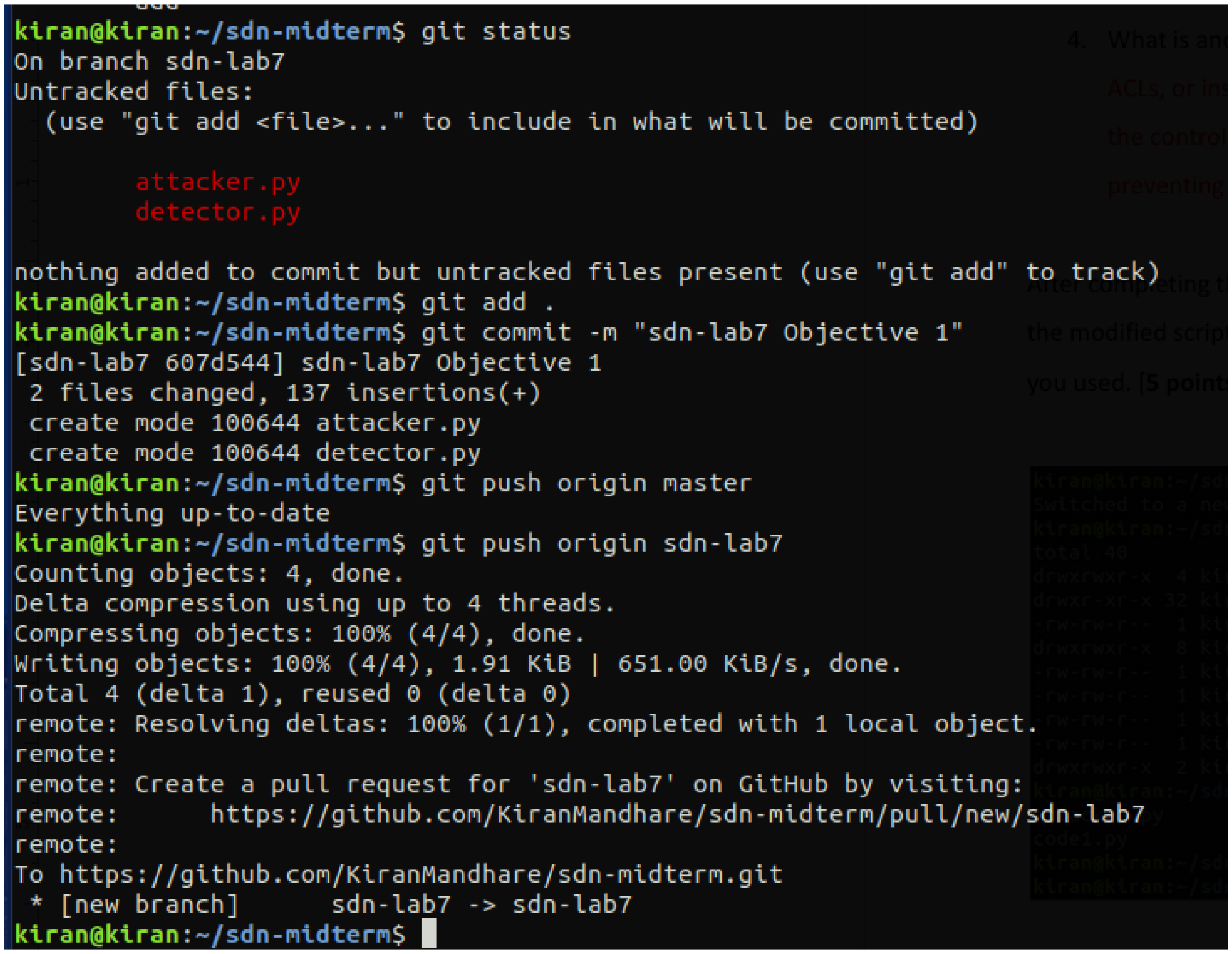
4. What is another way to prevent such attacks? [**1 point**]

   ACLs, or installing flows on the edges itself to block the traffic rather than blocking on the controller. The traffic can be blocked at the edge switch from where it originates, preventing it from even entering the network.

After completing the above objectives, create a new branch in your GitHub repo and then push the modified scripts back to GitHub making them master. Paste screenshots of the commands you used. [**5 points**]

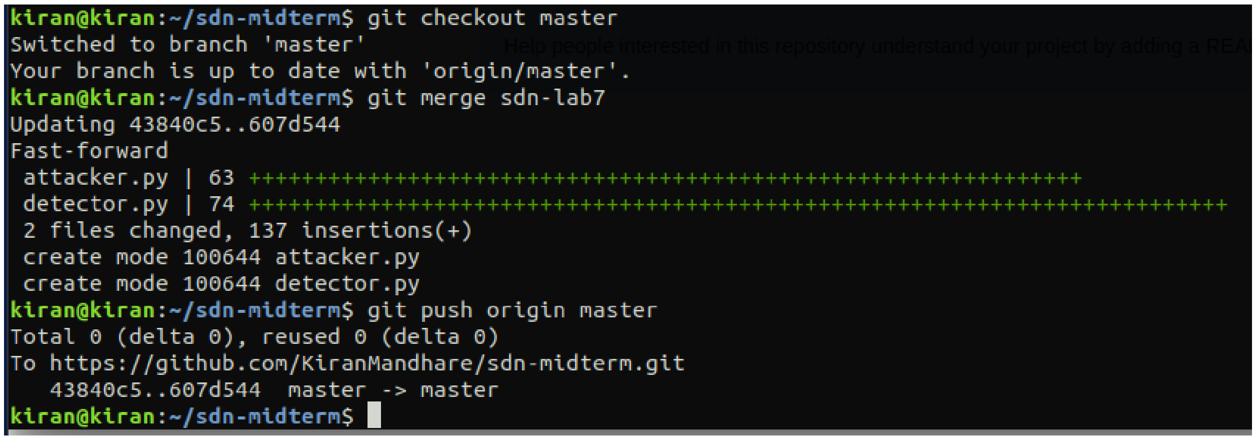


```
kiran@kiran:~/sdn-midterm$ git checkout -b sdn-lab7
Switched to a new branch 'sdn-lab7'
kiran@kiran:~/sdn-midterm$ ll
total 40
drwxrwxr-x  4 kiran kiran 4096 Oct 15 23:54 ./
drwxr-xr-x 32 kiran kiran 4096 Oct 27 11:28 ../
-rw-rw-r--  1 kiran kiran 6612 Oct 15 23:54 code1.py
drwxrwxr-x  8 kiran kiran 4096 Oct 27 13:12 .git/
-rw-rw-r--  1 kiran kiran   14 Oct 15 23:54 gitCode.py
-rw-rw-r--  1 kiran kiran  383 Oct 15 23:54 git-tester.py
-rw-rw-r--  1 kiran kiran 2003 Oct 15 23:54 pythonClient.py
-rw-rw-r--  1 kiran kiran  971 Oct 15 23:54 stats.py
drwxrwxr-x  2 kiran kiran 4096 Oct 15 23:54 templates/
kiran@kiran:~/sdn-midterm$ cp /home/kiran/sdn-lab7/sdn-midterm/
attacker.py             detector.py             git-tester.py           pythonClient.py
code1.py                gitCode.py              mininet@192.168.56.102  stats.py
kiran@kiran:~/sdn-midterm$ cp /home/kiran/sdn-lab7/sdn-midterm/attacker.py attacker.py
kiran@kiran:~/sdn-midterm$ cp /home/kiran/sdn-lab7/sdn-midterm/detector.py detector.py
```

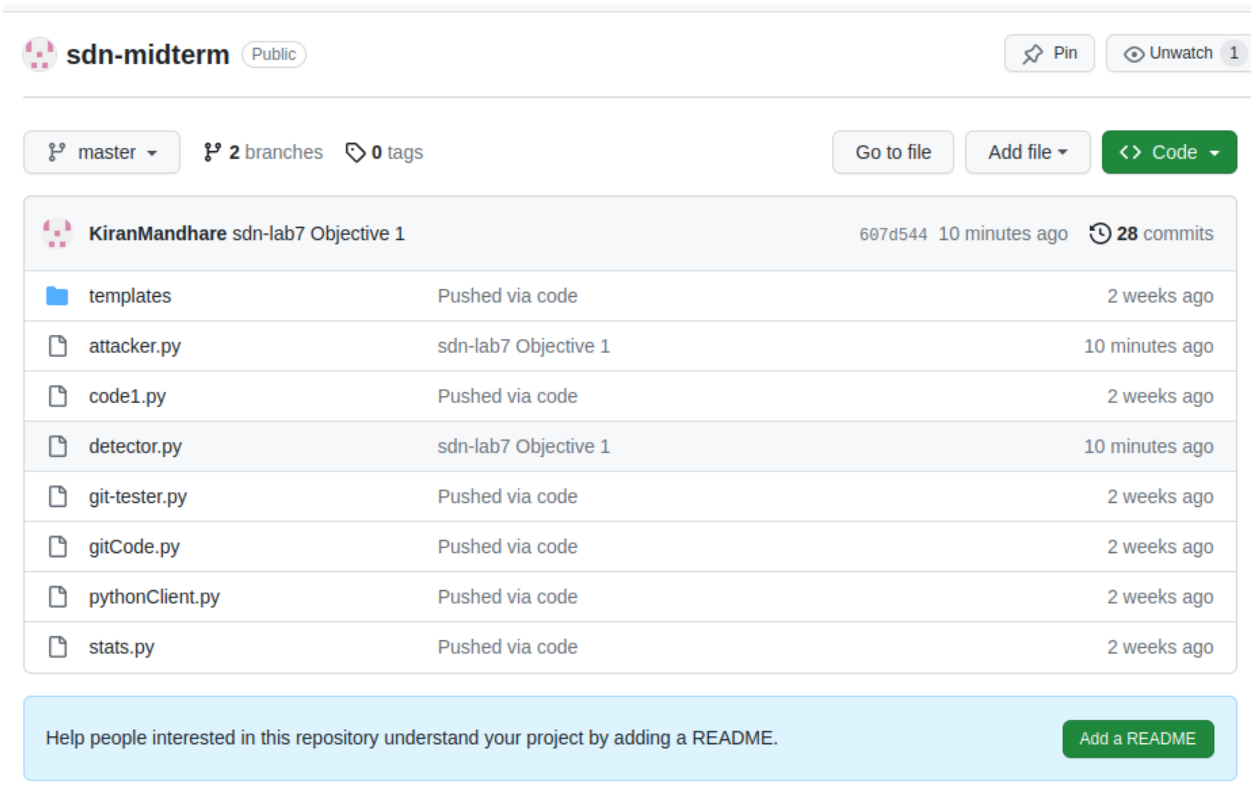


```
kiran@kiran:~/sdn-midterm$ git status
On branch sdn-lab7
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        attacker.py
        detector.py

nothing added to commit but untracked files present (use "git add" to track)
kiran@kiran:~/sdn-midterm$ git add .
kiran@kiran:~/sdn-midterm$ git commit -m "sdn-lab7 Objective 1"
[sdn-lab7 607d544] sdn-lab7 Objective 1
 2 files changed, 137 insertions(+)
 create mode 100644 attacker.py
 create mode 100644 detector.py
kiran@kiran:~/sdn-midterm$ git push origin master
Everything up-to-date
kiran@kiran:~/sdn-midterm$ git push origin sdn-lab7
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.91 KiB | 651.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'sdn-lab7' on GitHub by visiting:
remote:      https://github.com/KiranMandhare/sdn-midterm/pull/new/sdn-lab7
remote:
To https://github.com/KiranMandhare/sdn-midterm.git
 * [new branch]      sdn-lab7 -> sdn-lab7
kiran@kiran:~/sdn-midterm$
```

merge sdn-lab7 to master branch



```
kiran@kiran:~/sdn-midterm$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
kiran@kiran:~/sdn-midterm$ git merge sdn-lab7
Updating 43840c5..607d544
Fast-forward
 attacker.py | 63 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 detector.py | 74 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 137 insertions(+)
 create mode 100644 attacker.py
 create mode 100644 detector.py
kiran@kiran:~/sdn-midterm$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/KiranMandhare/sdn-midterm.git
   43840c5..607d544  master -> master
kiran@kiran:~/sdn-midterm$
```


sdn-midterm Public

master · 2 branches · 0 tags

KiranMandhare sdn-lab7 Objective 1 — 607d544 10 minutes ago — 28 commits

templates	Pushed via code	2 weeks ago
attacker.py	sdn-lab7 Objective 1	10 minutes ago
code1.py	Pushed via code	2 weeks ago
detector.py	sdn-lab7 Objective 1	10 minutes ago
git-tester.py	Pushed via code	2 weeks ago
gitCode.py	Pushed via code	2 weeks ago
pythonClient.py	Pushed via code	2 weeks ago
stats.py	Pushed via code	2 weeks ago

Help people interested in this repository understand your project by adding a README.

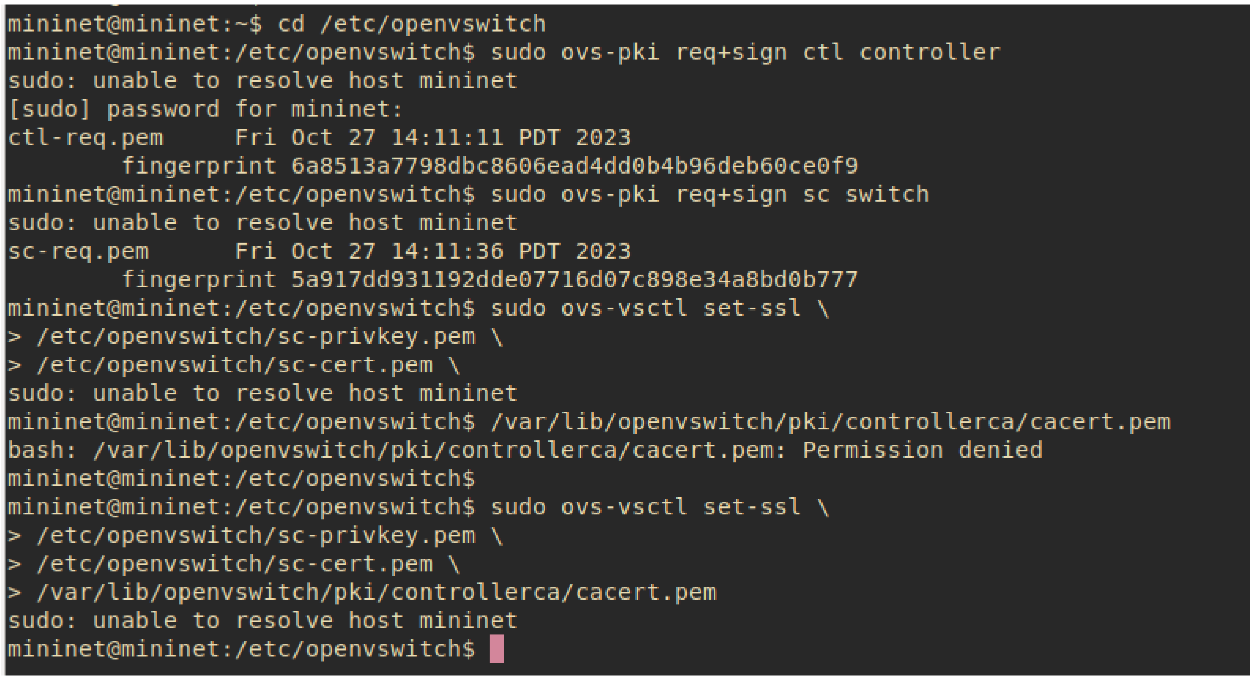
Objective 2 – SDN Security using SSL/TLS

Now that you have successfully attacked the controller and detected and stopped the attack, in this objective you will be creating SSL/TLS connections between Mininet and SDN controller. For simplicity you will only need to use Mininet VM to do this work. SSL/TLS is useful to secure SDN systems, but it cannot be enabled by a single command.

1. Run the following commands inside Mininet VM to generate all the keys required for this lab:

```
cd /etc/openvswitch
sudo ovs-pki req+sign ctl controller
sudo ovs-pki req+sign sc switch
sudo ovs-vsctl set-ssl \
        /etc/openvswitch/sc-privkey.pem \
        /etc/openvswitch/sc-cert.pem \
        /var/lib/openvswitch/pki/controllerca/cacert.pem
```

2. Provide screenshot of results. [**5 points**]



```
mininet@mininet:~$ cd /etc/openvswitch
mininet@mininet:/etc/openvswitch$ sudo ovs-pki req+sign ctl controller
sudo: unable to resolve host mininet
[sudo] password for mininet:
ctl-req.pem     Fri Oct 27 14:11:11 PDT 2023
        fingerprint 6a8513a7798dbc8606ead4dd0b4b96deb60ce0f9
mininet@mininet:/etc/openvswitch$ sudo ovs-pki req+sign sc switch
sudo: unable to resolve host mininet
sc-req.pem      Fri Oct 27 14:11:36 PDT 2023
        fingerprint 5a917dd931192dde07716d07c898e34a8bd0b777
mininet@mininet:/etc/openvswitch$ sudo ovs-vsctl set-ssl \
> /etc/openvswitch/sc-privkey.pem \
> /etc/openvswitch/sc-cert.pem \
sudo: unable to resolve host mininet
mininet@mininet:/etc/openvswitch$ /var/lib/openvswitch/pki/controllerca/cacert.pem
bash: /var/lib/openvswitch/pki/controllerca/cacert.pem: Permission denied
mininet@mininet:/etc/openvswitch$
mininet@mininet:/etc/openvswitch$ sudo ovs-vsctl set-ssl \
> /etc/openvswitch/sc-privkey.pem \
> /etc/openvswitch/sc-cert.pem \
> /var/lib/openvswitch/pki/controllerca/cacert.pem
sudo: unable to resolve host mininet
mininet@mininet:/etc/openvswitch$ 
```

3. Explain what is cert.pem, privkey,pem, req.pem. [**15 points**]

The file typically contains -

cert.pem -

The file contains the SSL/TLS certificate. The certificate is often in pem format and is a digital document signed by a trusted third party. The certificate binds the public key to an identity to be trusted by others.

This certificate is issued by a Certificate Authority and is used to establish the authenticity of a website or server, confirming that the server is indeed what it claims to be.

privkey.pem -

This is the key that is kept secret and is generated along with a req.pem for a specific domain/ website. This is mostly generated and kept safely in a vault. This key could give access to intruders over encrypted communication between the client and the server.

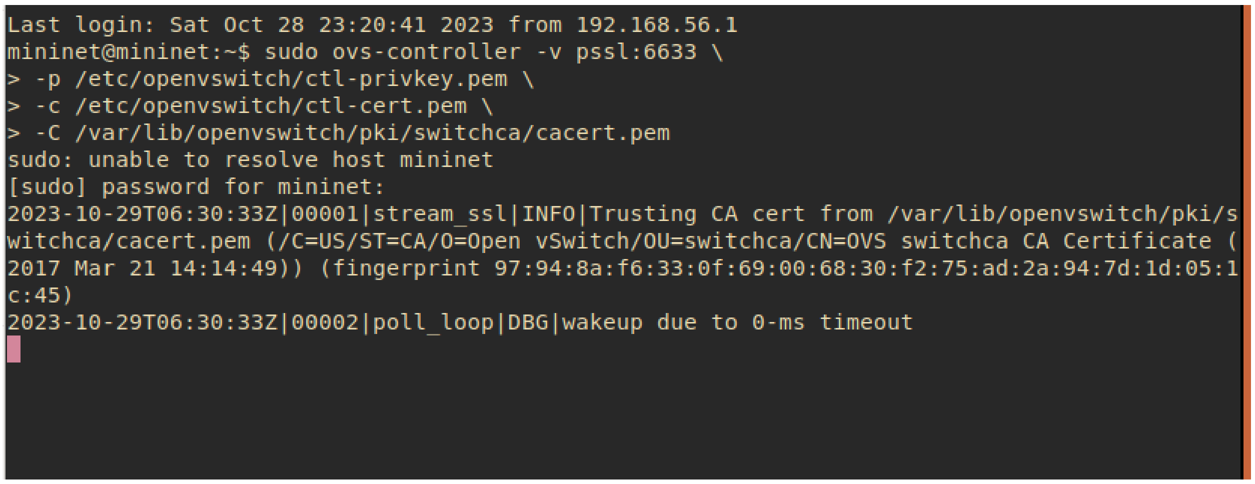
req.pem -

This file represents a certificate signing request. This key is provided to a third trusted party(Certificate Authority). Certificate Authority usually would sign this key and provide with a cer.pem which would be a chain of certs,signed by trusted entities and the domain which is signed.

4. In a separate window of the Mininet VM, run the following command:

   *sudo ovs-controller -v pssl:6633 *
   *-p /etc/openvswitch/ctl-privkey.pem *
   *-c /etc/openvswitch/ctl-cert.pem *
   -C /var/lib/openvswitch/pki/switchca/cacert.pem

   Explain what this command does and what the three options do. [**10 points**]



```
Last login: Sat Oct 28 23:20:41 2023 from 192.168.56.1
mininet@mininet:~$ sudo ovs-controller -v pssl:6633 \
> -p /etc/openvswitch/ctl-privkey.pem \
> -c /etc/openvswitch/ctl-cert.pem \
> -C /var/lib/openvswitch/pki/switchca/cacert.pem
sudo: unable to resolve host mininet
[sudo] password for mininet:
2023-10-29T06:30:33Z|00001|stream_ssl|INFO|Trusting CA cert from /var/lib/openvswitch/pki/s
witchca/cacert.pem (/C=US/ST=CA/O=Open vSwitch/OU=switchca/CN=OVS switchca CA Certificate (
2017 Mar 21 14:14:49)) (fingerprint 97:94:8a:f6:33:0f:69:00:68:30:f2:75:ad:2a:94:7d:1d:05:1
c:45)
2023-10-29T06:30:33Z|00002|poll_loop|DBG|wakeup due to 0-ms timeout
```

-v pssl:6633 ⟶

this flag specifies the protocol and port for the controller. In this case, it's using the SSL protocol on port 6633 fro secure communication.

-p /etc/openvswitch/ctl-privkey.pem ⟶

It specifies the path to the private key file used for SSL/TLS encryption. The private key is used to decrypt the data that is encrypted using the public key.

-c /etc/openvswitch/ctl-cert.pem —>

It points to the location of the SSL certificate file. The certificate contains information about the entity it identifies, along with the entity's public key.

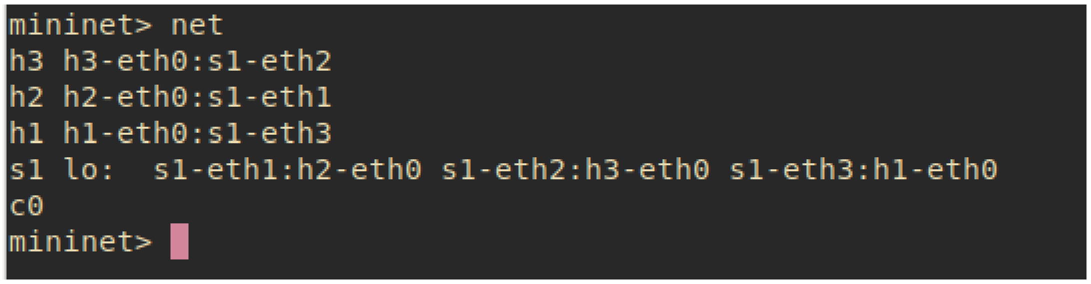
-C /var/lib/openvswitch/pki/switcha/cacert.pem —>

This refers to the path of the Certificate Authority (CA) certificate. The CA certificate validates the authenticity of the other certificates in the SSL communication, ensuring that they are from trusted sources, it has to be tracked to the root, the it also called trust chain.

This command starts the controller using SSL on port 6633 and provides the necessary SSL certificate and key files for secure communication between the controller and Open vSwitch.

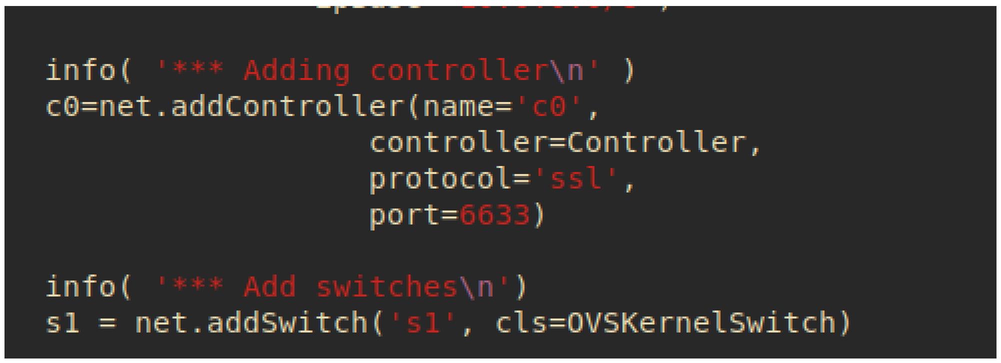
5. Create a basic topology with 1 controller, 1 switch and 3 hosts in MiniEdit. Export the .py file into the VM. Modify the .py file such that the OvS connects to the controller over a SSL connection. Paste screenshots of the topology and the modification to the .py file. [**15 points**]

Topology -



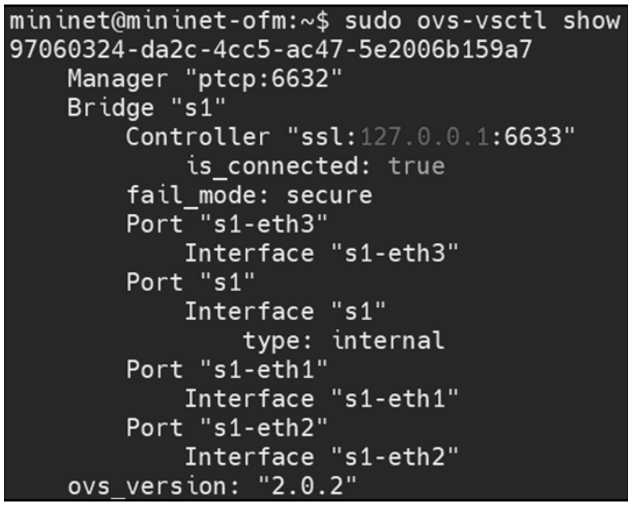
```
mininet> net
h3 h3-eth0:s1-eth2
h2 h2-eth0:s1-eth1
h1 h1-eth0:s1-eth3
s1 lo:  s1-eth1:h2-eth0 s1-eth2:h3-eth0 s1-eth3:h1-eth0
c0
mininet>
```

Code -



```
    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                      controller=Controller,
                      protocol='ssl',
                      port=6633)

    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
```

6. Execute this .py file and paste screenshot that indicates the switch is connected to the controller via a SSL secure connection. [**5 points**]



```
mininet@mininet-ofm:~$ sudo ovs-vsctl show
97060324-da2c-4cc5-ac47-5e2006b159a7
    Manager "ptcp:6632"
    Bridge "s1"
        Controller "ssl:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        Port "s1-eth3"
            Interface "s1-eth3"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.0.2"
```

7. Please describe the steps needed to create a SSL secure connection between a switch and a controller in your own words. [**5 points**]
    1) First we generate a key pair and a Certificate Signing request for a controller, with identity name as controller.
    2) Similarly, we generate a key pair and a CSR for a switch.
    3) We configure the CA certificate at the appropriate location for the controller to use and the public key is provided for use by the switch.
    4) The code now just needs to be configured to use an SSL certificate.

8. Can you describe the types of attack this objective can help prevent? [**5 points**]
    - **Man In Middle attack:** The SSL and TLS encryption prevents unauthorised interception of data between OpenvSwitch and controller. It enusures communication is encrypted, making it difficult for hackers to eavesdrop
    - **Spoofing Attacks:** By using SSL/TLS with certificates and keys, the controller and switch can authenticate each other. This prevents unauthorized devices or entities from impersonating the controller or the switch.

- **Unauthorized Acces to Network Configs:** Secure communication through SSL/TLS prevents unauthorized access or manipulation of the network configuration.

Objective 3 – SSL/TLS on Floodlight

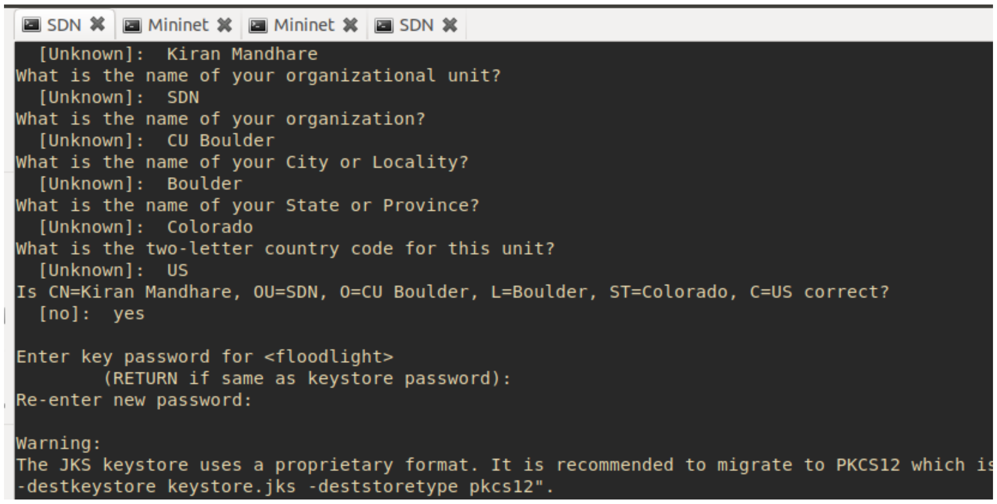
Complete the same objectives as in Obj 2 using Floodlight as the controller. Mention the steps you followed and paste screenshots indicating a successful SSL connection between OvS and the Floodlight controller. [**15 points**]

Steps -

1) Create keys and certificates on both switch and the controller -

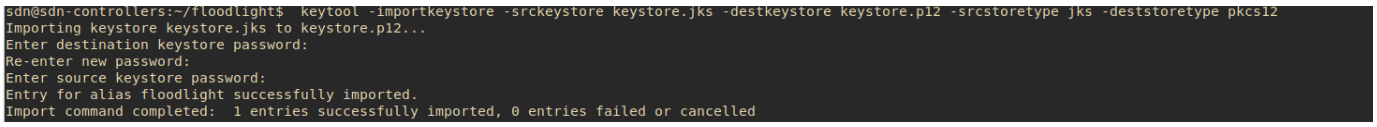
   On controller-

   keytool -genkey -keyalg RSA -alias floodlight -keystore keystore.jks -storepass changeit -validity 360 -keysize 2048



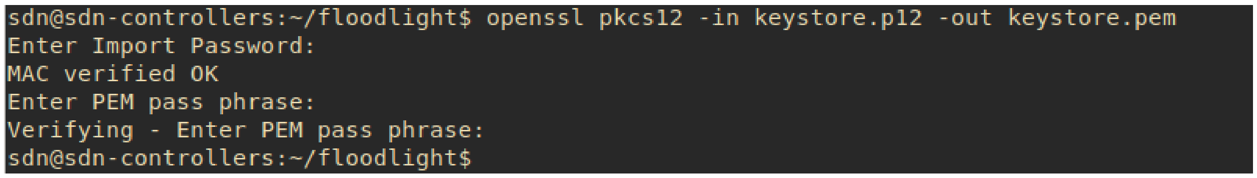
```
  [Unknown]:  Kiran Mandhare
What is the name of your organizational unit?
  [Unknown]:  SDN
What is the name of your organization?
  [Unknown]:  CU Boulder
What is the name of your City or Locality?
  [Unknown]:  Boulder
What is the name of your State or Province?
  [Unknown]:  Colorado
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Kiran Mandhare, OU=SDN, O=CU Boulder, L=Boulder, ST=Colorado, C=US correct?
  [no]:  yes

Enter key password for <floodlight>
        (RETURN if same as keystore password):
Re-enter new password:

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is
-destkeystore keystore.jks -deststoretype pkcs12".
```

Import the keys into keystore-



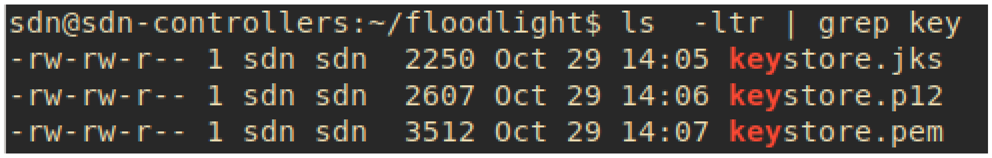
```
sdn@sdn-controllers:~/floodlight$  keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.p12 -srcstoretype jks -deststoretype pkcs12
Importing keystore keystore.jks to keystore.p12...
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
Entry for alias floodlight successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

Now, convert the PKCS12 file to a PEM file



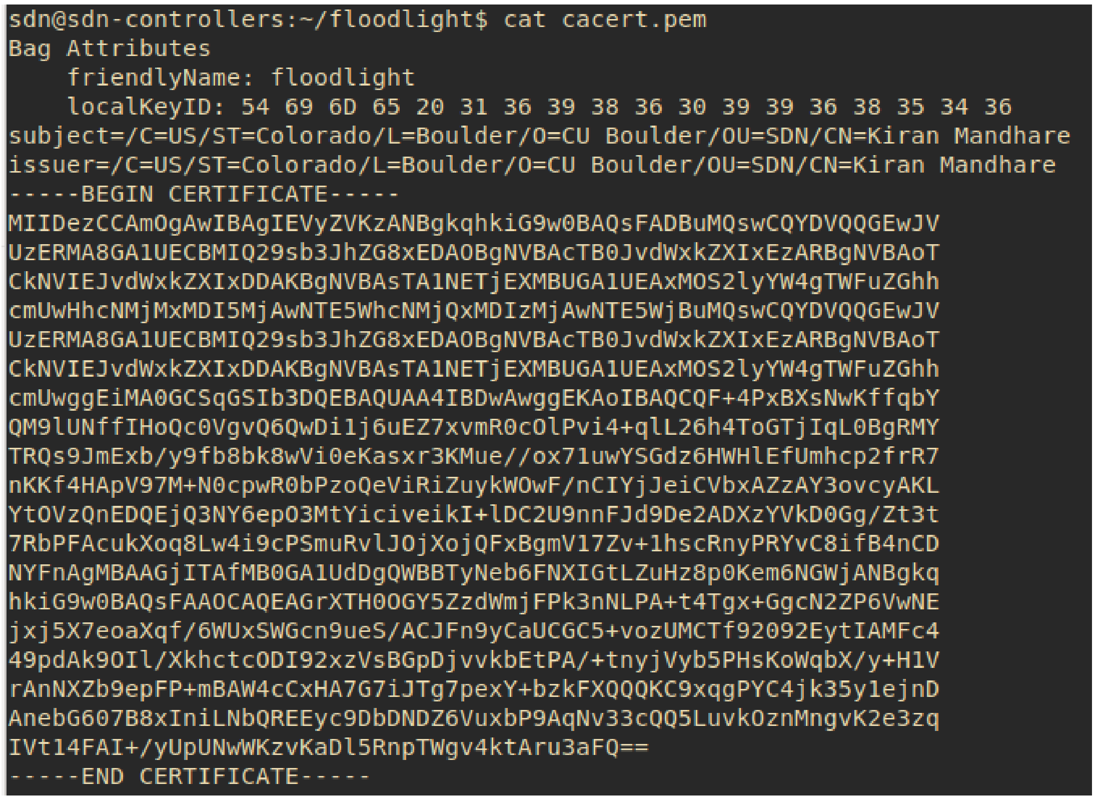
```
sdn@sdn-controllers:~/floodlight$ openssl pkcs12 -in keystore.p12 -out keystore.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
sdn@sdn-controllers:~/floodlight$
```

Following files are now present -



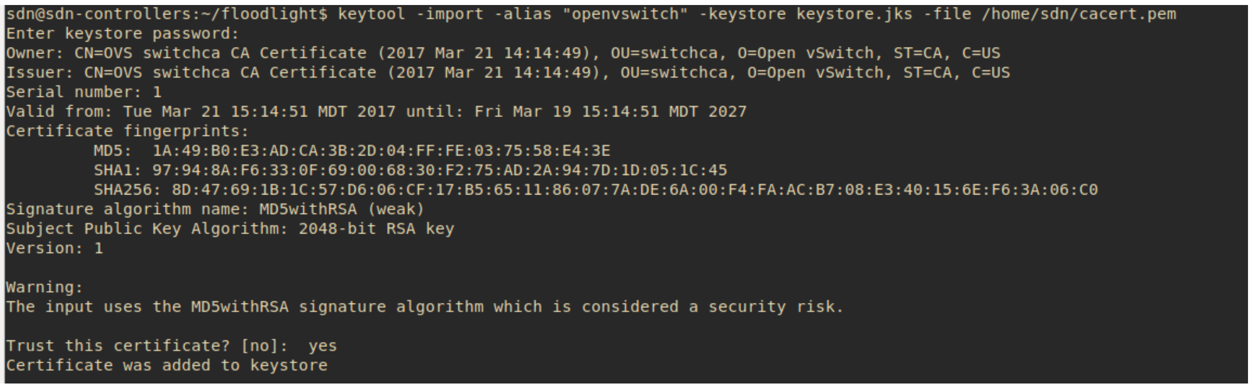
```
sdn@sdn-controllers:~/floodlight$ ls  -ltr | grep key
-rw-rw-r-- 1 sdn sdn  2250 Oct 29 14:05 keystore.jks
-rw-rw-r-- 1 sdn sdn  2607 Oct 29 14:06 keystore.p12
-rw-rw-r-- 1 sdn sdn  3512 Oct 29 14:07 keystore.pem
```

Extract the public key from keystore.pem and place it on openvswitch mininet VM for the switch to use it for handshake.



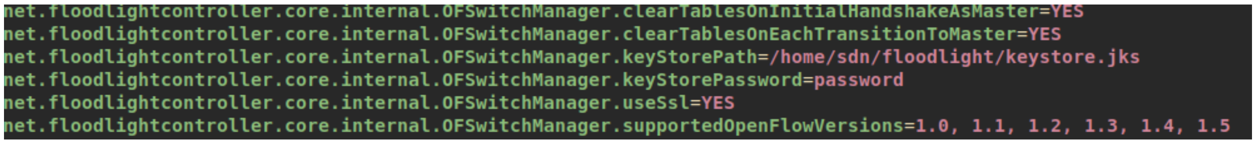
```
sdn@sdn-controllers:~/floodlight$ cat cacert.pem
Bag Attributes
    friendlyName: floodlight
    localKeyID: 54 69 6D 65 20 31 36 39 38 36 30 39 39 36 38 35 34 36
subject=/C=US/ST=Colorado/L=Boulder/O=CU Boulder/OU=SDN/CN=Kiran Mandhare
issuer=/C=US/ST=Colorado/L=Boulder/O=CU Boulder/OU=SDN/CN=Kiran Mandhare
-----BEGIN CERTIFICATE-----
MIIDezCCAmOgAwIBAgIEVyZVKzANBgkqhkiG9w0BAQsFADBuMQswCQYDVQQGEwJV
UzERMA8GA1UECBMIQ29sb3JhZG8xEDAOBgNVBAcTB0JvdWxkZXIxEzARBgNVBAoT
CkNVIEJvdWxkZXIxDDAKBgNVBAsTA1NETjEXMBUGA1UEAxMOS2lyYW4gTWFuZGhh
cmUwHhcNMjMxMDI5MjAwNTE5WhcNMjQxMDIzMjAwNTE5WjBuMQswCQYDVQQGEwJV
UzERMA8GA1UECBMIQ29sb3JhZG8xEDAOBgNVBAcTB0JvdWxkZXIxEzARBgNVBAoT
CkNVIEJvdWxkZXIxDDAKBgNVBAsTA1NETjEXMBUGA1UEAxMOS2lyYW4gTWFuZGhh
cmUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCQF+4PxBXsNwKffqbY
QM9lUNffIHoQc0VgvQ6QwDi1j6uEZ7xvmR0cOlPvi4+qlL26h4ToGTjIqL0BgRMY
TRQs9JmExb/y9fb8bk8wVi0eKasxr3KMue//ox71uwYSGdz6HWHlEfUmhcp2frR7
nKKf4HApV97M+N0cpwR0bPzoQeViRiZuykWOwF/nCIYjJeiCVbxAZzAY3ovcyAKL
YtOVzQnEDQEjQ3NY6epO3MtYiciveikI+lDC2U9nnFJd9De2ADXzYVkD0Gg/Zt3t
7RbPFAcukXoq8Lw4i9cPSmuRvlJOjXojQFxBgmV17Zv+1hscRnyPRYvC8ifB4nCD
NYFnAgMBAAGjITAfMB0GA1UdDgQWBBTyNeb6FNXIGtLZuHz8p0Kem6NGWjANBgkq
hkiG9w0BAQsFAAOCAQEAGrXTH0OGY5ZzdWmjFPk3nNLPA+t4Tgx+GgcN2ZP6VwNE
jxj5X7eoaXqf/6WUxSWGcn9ueS/ACJFn9yCaUCGC5+vozUMCTf92092EytIAMFc4
49pdAk9OIl/XkhctcODI92xzVsBGpDjvvkbEtPA/+tnyjVyb5PHsKoWqbX/y+H1V
rAnNXZb9epFP+mBAW4cCxHA7G7iJTg7pexY+bzkFXQQQKC9xqgPYC4jk35y1ejnD
AnebG607B8xIniLNbQREEyc9DbDNDZ6VuxbP9AqNv33cQQ5LuvkOznMngvK2e3zq
IVt14FAI+/yUpUNwWKzvKaDl5RnpTWgv4ktAru3aFQ==
-----END CERTIFICATE-----
```

Repeat similar steps on the ovs-switch and export the cacert.pem and add it to keystore on SDN vm.



```
sdn@sdn-controllers:~/floodlight$ keytool -import -alias "openvswitch" -keystore keystore.jks -file /home/sdn/cacert.pem
Enter keystore password:
Owner: CN=OVS switchca CA Certificate (2017 Mar 21 14:14:49), OU=switchca, O=Open vSwitch, ST=CA, C=US
Issuer: CN=OVS switchca CA Certificate (2017 Mar 21 14:14:49), OU=switchca, O=Open vSwitch, ST=CA, C=US
Serial number: 1
Valid from: Tue Mar 21 15:14:51 MDT 2017 until: Fri Mar 19 15:14:51 MDT 2027
Certificate fingerprints:
         MD5:  1A:49:B0:E3:AD:CA:3B:2D:04:FF:FE:03:75:58:E4:3E
         SHA1: 97:94:8A:F6:33:0F:69:00:68:30:F2:75:AD:2A:94:7D:1D:05:1C:45
         SHA256: 8D:47:69:1B:1C:57:D6:06:CF:17:B5:65:11:86:07:7A:DE:6A:00:F4:FA:AC:B7:08:E3:40:15:6E:F6:3A:06:C0
Signature algorithm name: MD5withRSA (weak)
Subject Public Key Algorithm: 2048-bit RSA key
Version: 1

Warning:
The input uses the MD5withRSA signature algorithm which is considered a security risk.

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

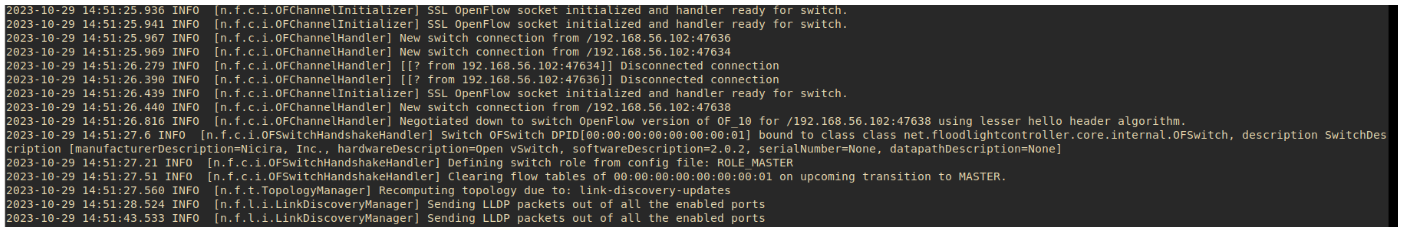
In floodlightdefault.properities file enable SSL and provide information about cert store -key store



```
net.floodlightcontroller.core.internal.OFSwitchManager.clearTablesOnInitialHandshakeAsMaster=YES
net.floodlightcontroller.core.internal.OFSwitchManager.clearTablesOnEachTransitionToMaster=YES
net.floodlightcontroller.core.internal.OFSwitchManager.keyStorePath=/home/sdn/floodlight/keystore.jks
net.floodlightcontroller.core.internal.OFSwitchManager.keyStorePassword=password
net.floodlightcontroller.core.internal.OFSwitchManager.useSsl=YES
net.floodlightcontroller.core.internal.OFSwitchManager.supportedOpenFlowVersions=1.0, 1.1, 1.2, 1.3, 1.4, 1.5
```

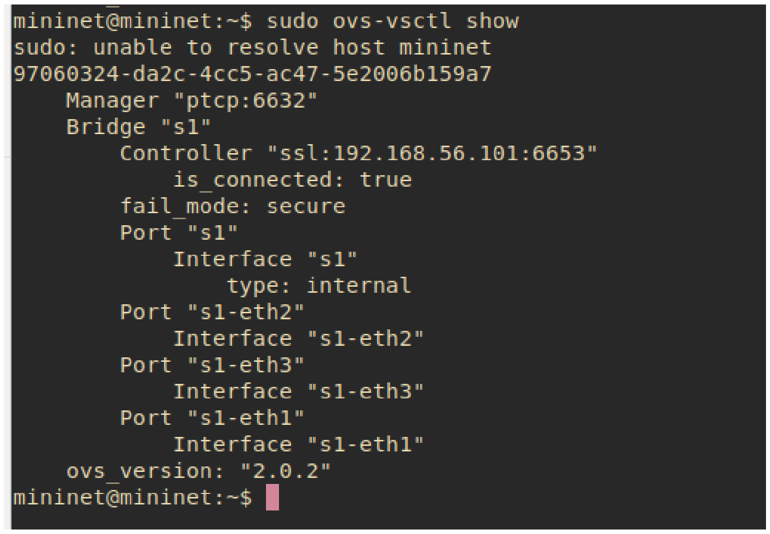
Start the Floodlight controller and start mininet topology

Controller logs -



```
2023-10-29 14:51:25.936 INFO  [n.f.c.i.OFChannelInitializer] SSL OpenFlow socket initialized and handler ready for switch.
2023-10-29 14:51:25.941 INFO  [n.f.c.i.OFChannelInitializer] SSL OpenFlow socket initialized and handler ready for switch.
2023-10-29 14:51:25.967 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /192.168.56.102:47636
2023-10-29 14:51:25.969 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /192.168.56.102:47634
2023-10-29 14:51:26.279 INFO  [n.f.c.i.OFChannelHandler] [[? from 192.168.56.102:47634]] Disconnected connection
2023-10-29 14:51:26.390 INFO  [n.f.c.i.OFChannelHandler] [[? from 192.168.56.102:47636]] Disconnected connection
2023-10-29 14:51:26.439 INFO  [n.f.c.i.OFChannelInitializer] SSL OpenFlow socket initialized and handler ready for switch.
2023-10-29 14:51:26.440 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /192.168.56.102:47638
2023-10-29 14:51:26.816 INFO  [n.f.c.i.OFChannelHandler] Negotiated down to switch OpenFlow version of OF_10 for /192.168.56.102:47638 using lesser hello header algorithm.
2023-10-29 14:51:27.6 INFO  [n.f.c.i.OFSwitchHandshakeHandler] Switch OFSwitch DPID[00:00:00:00:00:00:00:01] bound to class class net.floodlightcontroller.core.internal.OFSwitch, description SwitchDes
cription [manufacturerDescription=Nicira, Inc., hardwareDescription=Open vSwitch, softwareDescription=2.0.2, serialNumber=None, datapathDescription=None]
2023-10-29 14:51:27.21 INFO  [n.f.c.i.OFSwitchHandshakeHandler] Defining switch role from config file: ROLE_MASTER
2023-10-29 14:51:27.51 INFO  [n.f.c.i.OFSwitchHandshakeHandler] Clearing flow tables of 00:00:00:00:00:00:00:01 on upcoming transition to MASTER.
2023-10-29 14:51:27.560 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-10-29 14:51:28.524 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-10-29 14:51:43.533 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
```

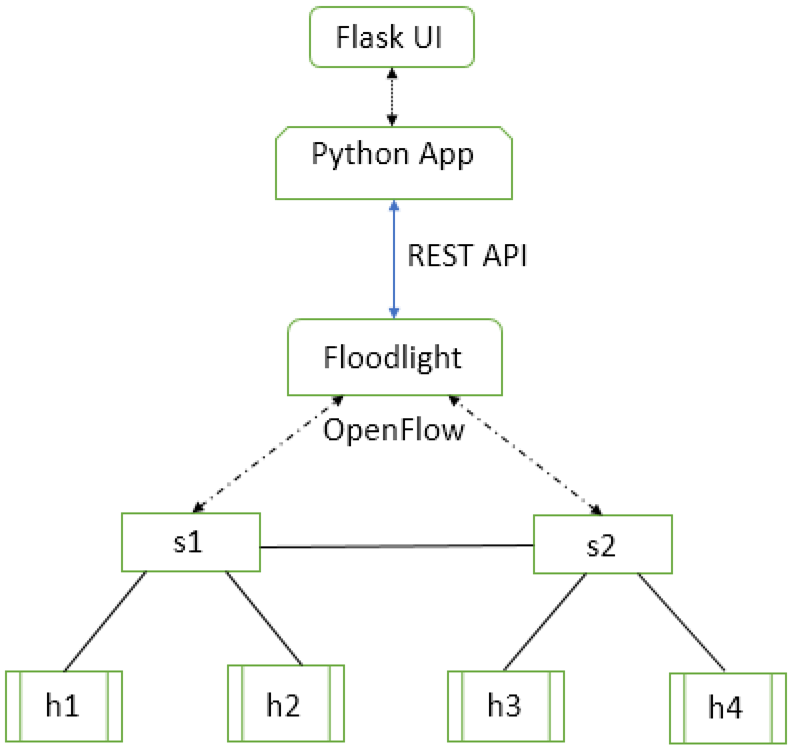
Mininet connects to floodlight on SSL connection on port 6653



```
mininet@mininet:~$ sudo ovs-vsctl show
sudo: unable to resolve host mininet
97060324-da2c-4cc5-ac47-5e2006b159a7
    Manager "ptcp:6632"
    Bridge "s1"
        Controller "ssl:192.168.56.101:6653"
            is_connected: true
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth3"
            Interface "s1-eth3"
        Port "s1-eth1"
            Interface "s1-eth1"
    ovs_version: "2.0.2"
mininet@mininet:~$
```

Objective 4 – REST Static flow entries and Firewall via Python

In this objective, you will be writing a Python script that uses Flask (or any library of choice, but “Flask” will be used throughout this document) to create a GUI, takes inputs from the user via the GUI and uses the REST API on Floodlight to configure static flow entries and firewall rules.



1. Before initializing Floodlight, edit the file - /home/sdn/floodlight/src/main/resources/ floodlightdefault.properties file and remove the line - net.floodlightcontroller.forwarding.Forwarding

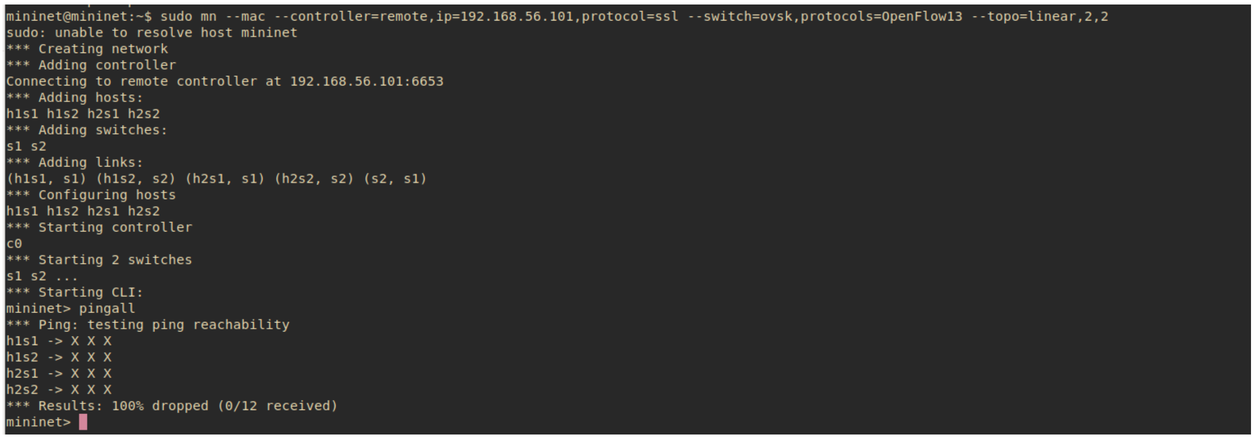
   Can you explain what removing this line does? [**5 points**]

   It is the Forwarding App, for L2 domain that basically learns switch mac and installs flows onto the switch for forwarding operation based on the known MAC addresses. By removing this line, there is no way the switch App is running on the controller and hence the flows won't be installed for forwarding packets. Basically transparent behavior of the switch won't happen.

   Flows explicitly need to be added on to the ovsk for forwarding to work.

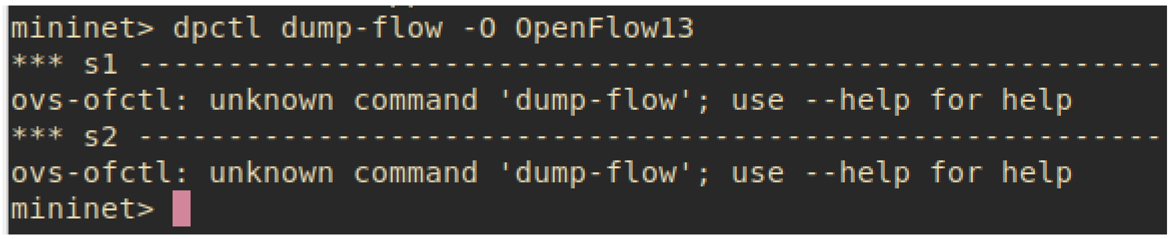
2. Create the above topology in Mininet with two switches and two hosts connected to each switch and the remote Floodlight controller. Do a pingall.

   Will it work? Why/why not? Do you see flow entries on the switches? [**5 points**]



```
mininet@mininet:~$ sudo mn --mac --controller=remote,ip=192.168.56.101,protocol=ssl --switch=ovsk,protocols=OpenFlow13 --topo=linear,2,2
sudo: unable to resolve host mininet
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.56.101:6653
*** Adding hosts:
h1s1 h1s2 h2s1 h2s2
*** Adding switches:
s1 s2
*** Adding links:
(h1s1, s1) (h1s2, s2) (h2s1, s1) (h2s2, s2) (s2, s1)
*** Configuring hosts
h1s1 h1s2 h2s1 h2s2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> X X X
h1s2 -> X X X
h2s1 -> X X X
h2s2 -> X X X
*** Results: 100% dropped (0/12 received)
mininet>
```

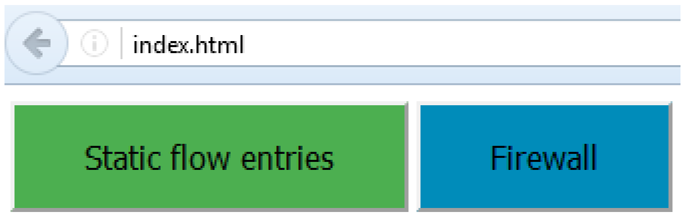
It doesn't work because the Forwarding module isn't running on the controller. Thus regular switching -learning, forwarding won't happen.

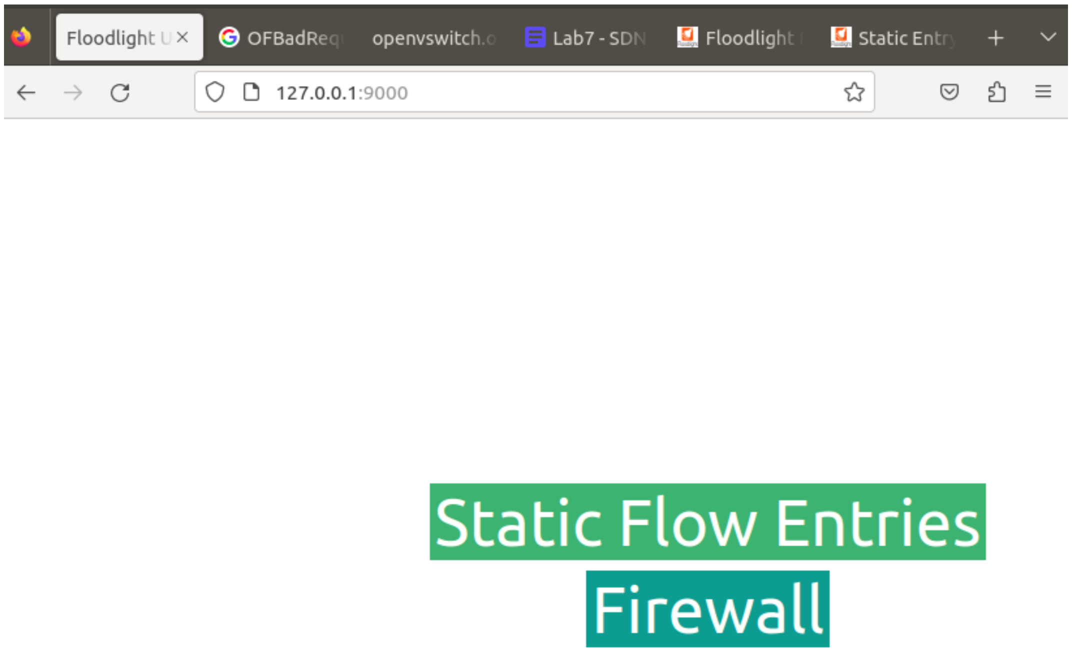


```
mininet> dpctl dump-flow -O OpenFlow13
*** s1 ------------------------------------------------------------
ovs-ofctl: unknown command 'dump-flow'; use --help for help
*** s2 ------------------------------------------------------------
ovs-ofctl: unknown command 'dump-flow'; use --help for help
mininet>
```

There are no flows on the switch.

3. Write a Python script to create a simple REST client for accessing the controller's REST API.
4. Use Flask to create a GUI, index.html should display 2 options – Static Routing and Firewall, and depending on what user selects redirect to another appropriate html page.





Static Flow Entries

Firewall

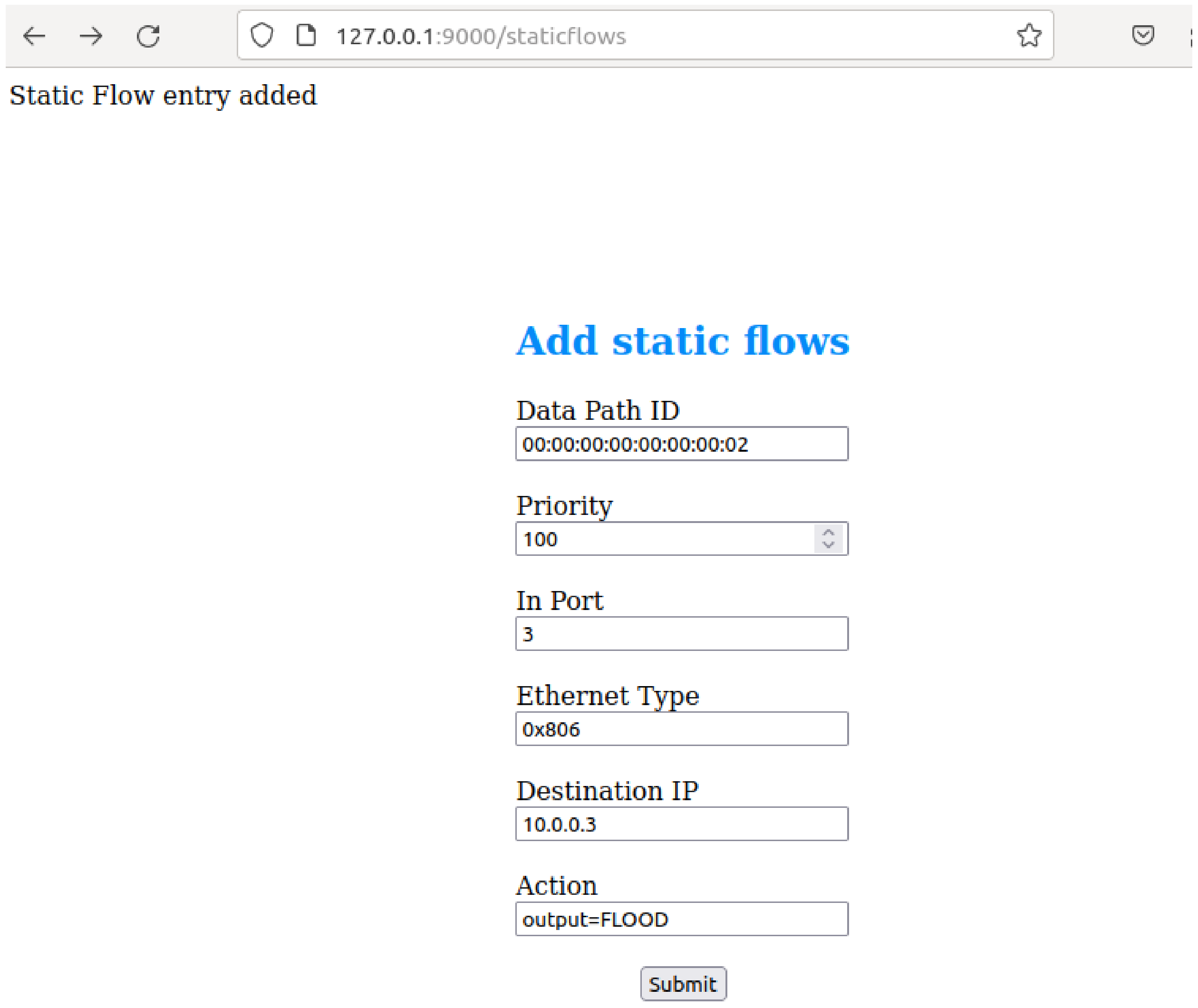
5. The static routing html page should take inputs from the user for these fields:

   DPID, priority, In-Port, Eth-type, Dest IP, Action (flood or the particular port number)

   You will have to add static flow entries on switches s1 and s2 to flood ARP packets, and forward other packets to the appropriate out port so that all hosts are able to ping each other.

   Paste screenshots of the relevant flow entries on the switches. [**5 points**]

WebPage



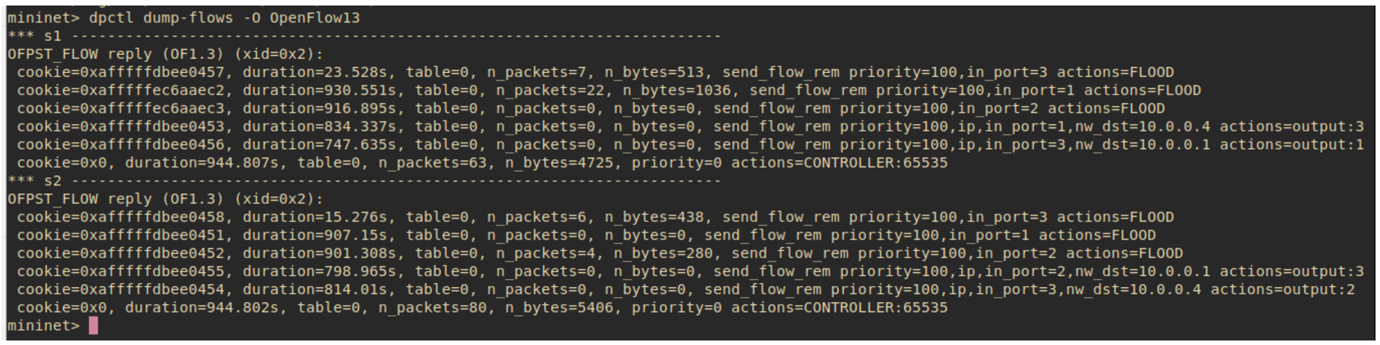
127.0.0.1:9000/staticflows

Static Flow entry added

Add static flows

Data Path ID	00:00:00:00:00:00:00:02
Priority	100
In Port	3
Ethernet Type	0x806
Destination IP	10.0.0.3
Action	output=FLOOD

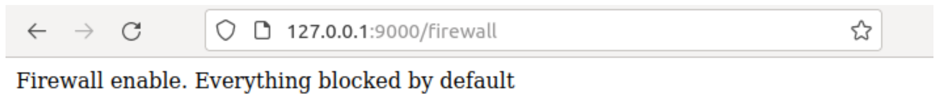
Following flows are added via Flask UI



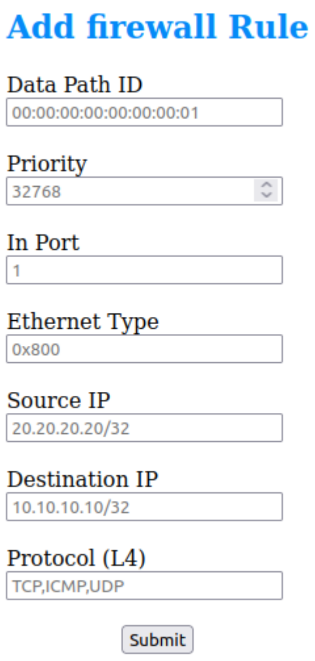
```
mininet> dpctl dump-flows -O OpenFlow13
*** s1 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0xaffffffdbee0457, duration=23.528s, table=0, n_packets=7, n_bytes=513, send_flow_rem priority=100,in_port=3 actions=FLOOD
 cookie=0xafffffec6aaec2, duration=930.551s, table=0, n_packets=22, n_bytes=1036, send_flow_rem priority=100,in_port=1 actions=FLOOD
 cookie=0xafffffec6aaec3, duration=916.895s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,in_port=2 actions=FLOOD
 cookie=0xaffffffdbee0453, duration=834.337s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,ip,in_port=1,nw_dst=10.0.0.4 actions=output:3
 cookie=0xaffffffdbee0456, duration=747.635s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,ip,in_port=3,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=944.807s, table=0, n_packets=63, n_bytes=4725, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0xaffffffdbee0458, duration=15.276s, table=0, n_packets=6, n_bytes=438, send_flow_rem priority=100,in_port=3 actions=FLOOD
 cookie=0xaffffffdbee0451, duration=907.15s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,in_port=1 actions=FLOOD
 cookie=0xaffffffdbee0452, duration=901.308s, table=0, n_packets=4, n_bytes=280, send_flow_rem priority=100,in_port=2 actions=FLOOD
 cookie=0xaffffffdbee0455, duration=798.965s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,ip,in_port=2,nw_dst=10.0.0.1 actions=output:3
 cookie=0xaffffffdbee0454, duration=814.01s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,ip,in_port=3,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=944.802s, table=0, n_packets=80, n_bytes=5406, priority=0 actions=CONTROLLER:65535
mininet> 
```

6. When the user selects Firewall, by default everything should be blocked.
7. The firewall html page should take inputs from the user for these fields:
   DPID, priority, In-Port, Eth-type, Src IP, Dest IP, L4 protocol
   You will have to add firewall rules to allow specific communication based on the user inputs. Do not use the Firewall API, add static flow entries for the traffic allowed.
8. Paste screenshots of the firewall rule added and the ping outputs indicating only the specific connections allowed.

WebPage -Firewall


127.0.0.1:9000/firewall

Firewall enable. Everything blocked by default


Add firewall Rule

Data Path ID
00:00:00:00:00:00:00:01

Priority
32768

In Port
1

Ethernet Type
0x800

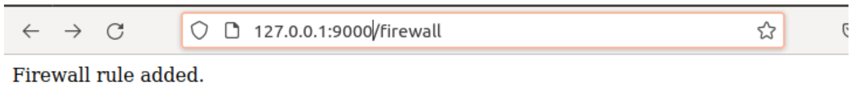
Source IP
20.20.20.20/32

Destination IP
10.10.10.10/32

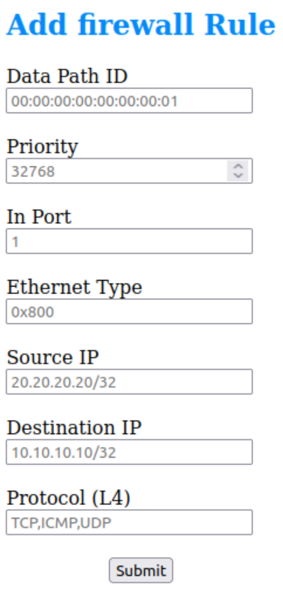
Protocol (L4)
TCP,ICMP,UDP

By selecting the firewall button everything is disabled, all flows are removed from the switch by default when the firewall is disabled for the first time.
Firewall page allows to add entries to allow specific traffic.

Since, the closest action would be FLOOD for ALLOW rule in static flow pusher and hence entries that allow specific traffic would be marked as flood.


127.0.0.1:9000/firewall

Firewall rule added.


Add firewall Rule

Data Path ID: 00:00:00:00:00:00:00:01

Priority: 32768

In Port: 1

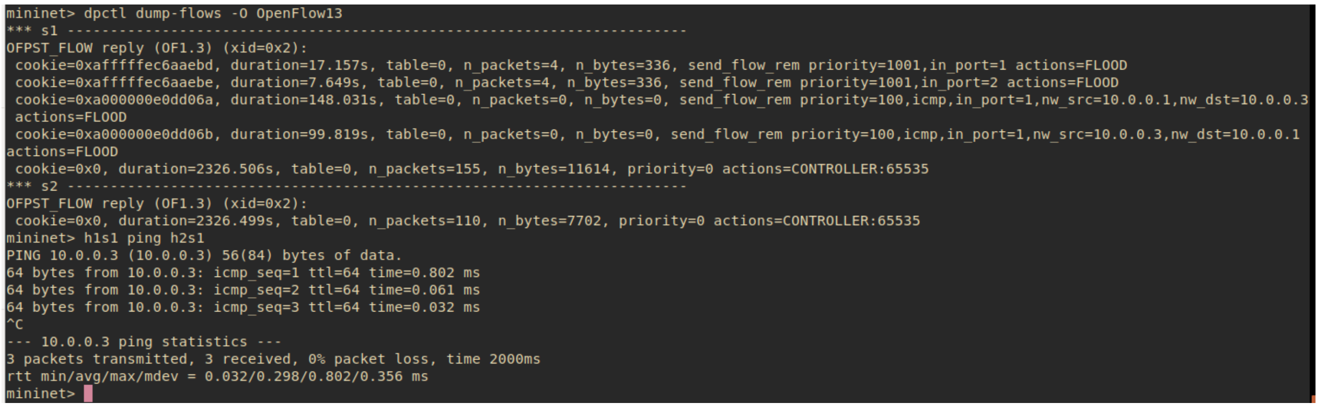
Ethernet Type: 0x800

Source IP: 20.20.20.20/32

Destination IP: 10.10.10.10/32

Protocol (L4): TCP,ICMP,UDP

Flows and Ping output


```
mininet> dpctl dump-flows -O OpenFlow13
*** s1 -----------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0xafffffec6aaebd, duration=17.157s, table=0, n_packets=4, n_bytes=336, send_flow_rem priority=1001,in_port=1 actions=FLOOD
 cookie=0xafffffec6aaebe, duration=7.649s, table=0, n_packets=4, n_bytes=336, send_flow_rem priority=1001,in_port=2 actions=FLOOD
 cookie=0xa000000e0dd06a, duration=148.031s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,icmp,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.3
 actions=FLOOD
 cookie=0xa000000e0dd06b, duration=99.819s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=100,icmp,in_port=1,nw_src=10.0.0.3,nw_dst=10.0.0.1
actions=FLOOD
 cookie=0x0, duration=2326.506s, table=0, n_packets=155, n_bytes=11614, priority=0 actions=CONTROLLER:65535
*** s2 -----------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=2326.499s, table=0, n_packets=110, n_bytes=7702, priority=0 actions=CONTROLLER:65535
mininet> h1s1 ping h2s1
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.802 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.032 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.032/0.298/0.802/0.356 ms
mininet> 
```

9. To achieve full credit, attach the script along your submission and please show the functioning of your code to the TAs. [**80 points**]

Total Points ___ / 237 points