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

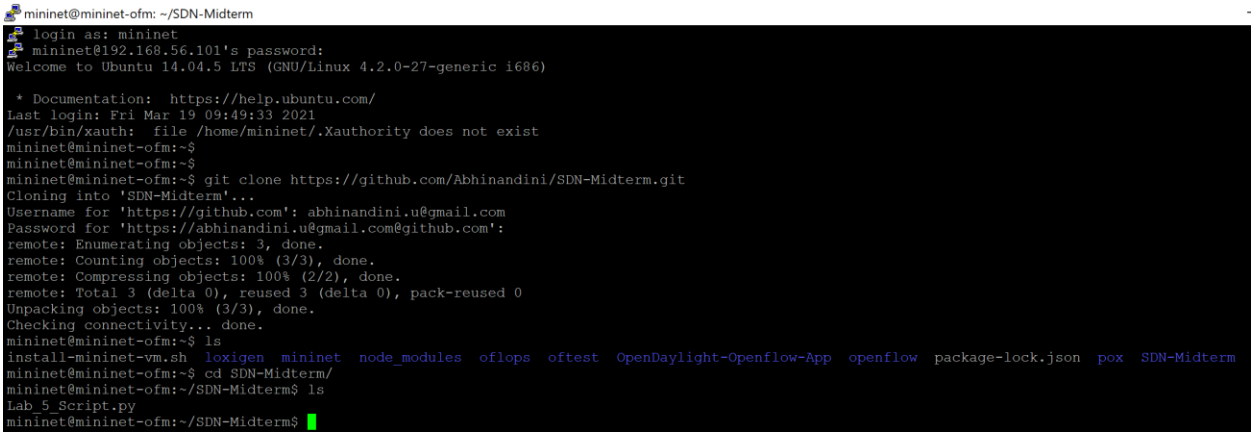
This lab is intended to be an overview of basic TLS and firewall security in SDN. Securing the SDN controller is critical to the security of the entire SDN. TLS is the standard way to secure the southbound communication from the networking devices and the SDN controller. This lab will provide the fundamentals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

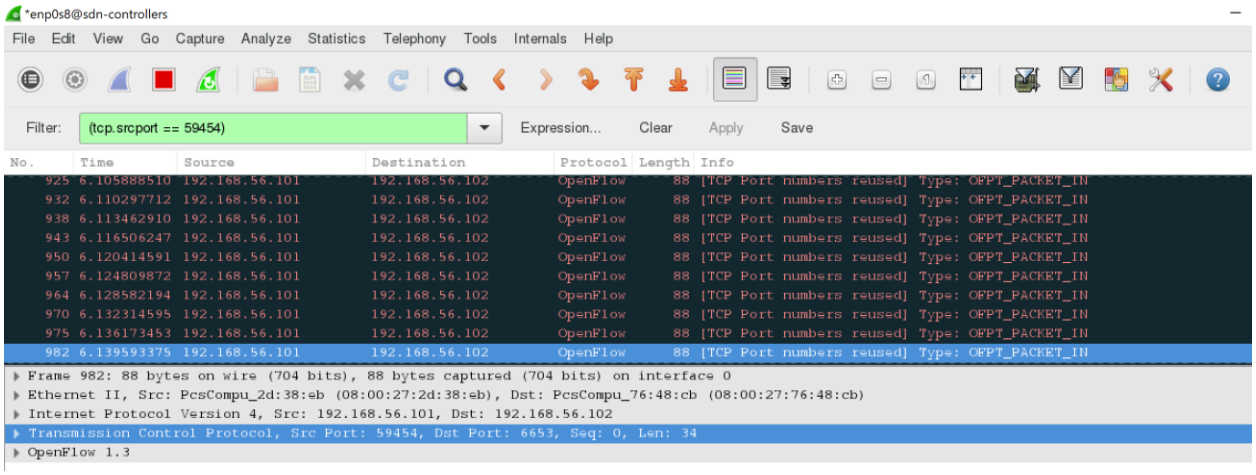
1. Initialize Floodlight on the controllers VM.
2. Initialize a linear topology in Mininet with four switches, remote Floodlight controller, and OpenFlow v1.3.
3. Clone the scripts from your GitHub account you pushed for Lab 5 on the Mininet VM. Paste a screenshot of the commands used. [**1 point**]



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 detect 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**]

```
            is_connected: true
        fail_mode: secure
        Port "s3-eth3"
            Interface "s3-eth3"
        Port "s3-eth2"
            Interface "s3-eth2"
        Port "s3-eth1"
            Interface "s3-eth1"
        Port "s3"
            Interface "s3"
                type: internal
    Bridge "s4"
        Controller "tcp:192.168.56.102:6653"
            is_connected: true
        Controller "ptcp:6657"
        fail_mode: secure
        Port "s4"
            Interface "s4"
                type: internal
        Port "s4-eth1"
            Interface "s4-eth1"
        Port "s4-eth2"
            Interface "s4-eth2"
    ovs_version: "2.0.2"

 Enter the Controller IP: 192.168.56.102

 Enter the Switch Port: 59454

 Enter the Controller Port: 6653
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```



Detect and stop:

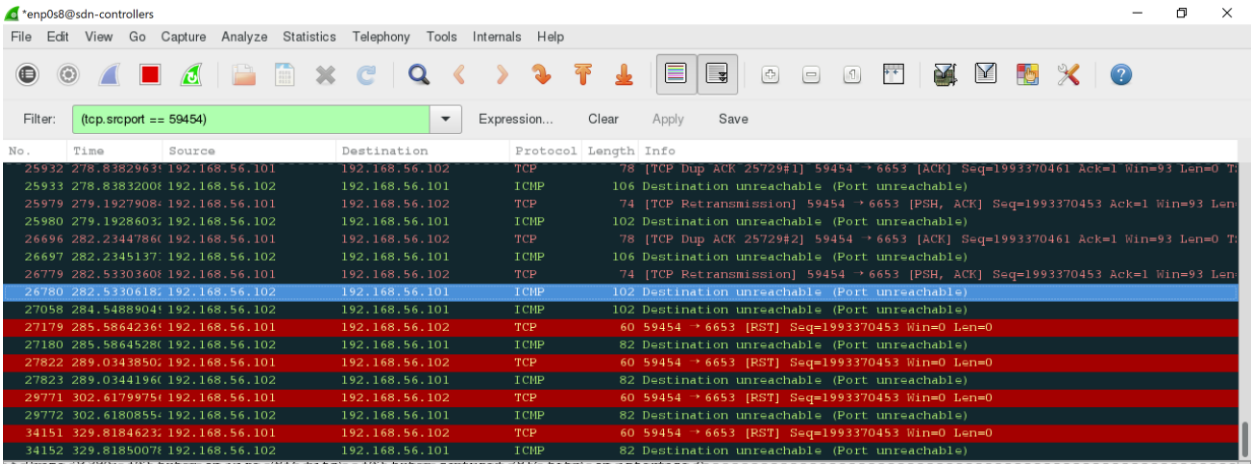
1. To detect the attack, use the script from Lab 5 to count the number of Packet_In messages received from a switch IP:port.
2. You must execute the script on the controllers VM and display a message when a threshold (say more than 100 Packet_In messages from one switch IP:port) are detected. Paste screenshots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

```
import subprocess
import shlex
import time


output = subprocess.check_output(shlex.split("""sudo tshark -f "(host 192.168.56.101 and tcp port 59454)" -i enp0s8 -d tcp.port==6653,openflow -O openflow_v4 -Y "((openflow_v4.type == 10))" -c 100"""))
print ("\nThreshold Reached")
```

```
sdn@sdn-controllers:~$ sudo python scapy_detect.py
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s8'
73

Threshold Reached
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  192.168.56.101       192.168.56.102       tcp spt:59454 dpt:6653 reject-with icmp-port-unreachable
sdn@sdn-controllers:~$
```

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 python scapy_detect.py
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s8'
73

Threshold Reached
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  192.168.56.101       192.168.56.102       tcp spt:59454 dpt:6653 reject-with icmp-port-unreachable
sdn@sdn-controllers:~$
```



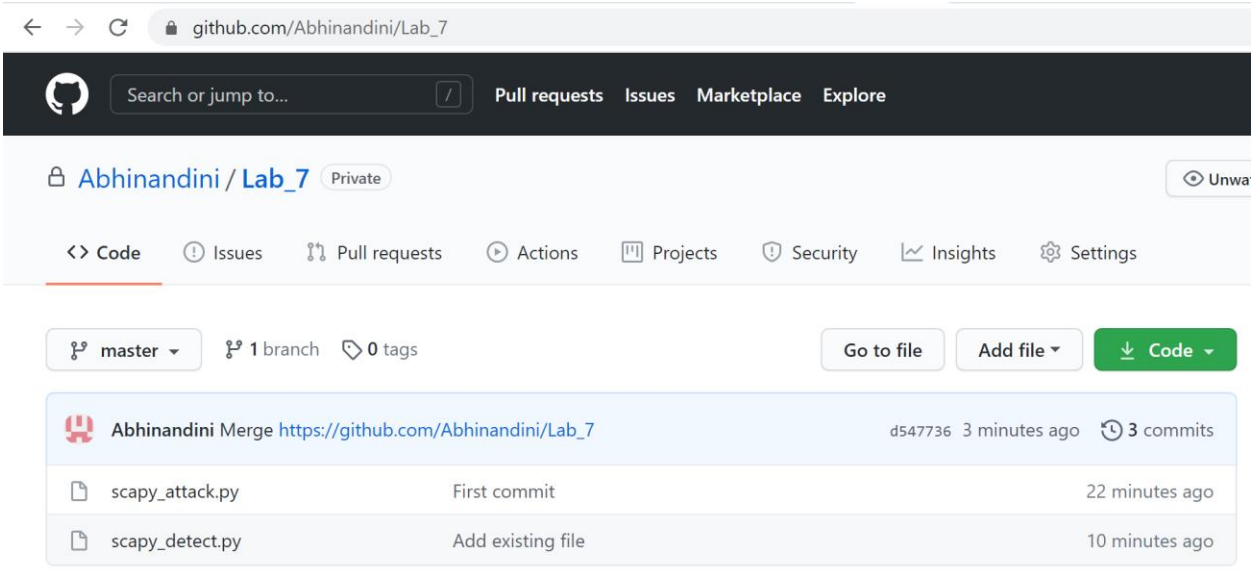
4. What is another way to prevent such attacks? [**1 point**]

Attack can be prevented by implementing entropy based detection and mitigation algorithm or by connecting to controller with TLS.

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**]

```
mininet@mininet-ofm:~/mininet/custom$ git init
Initialized empty Git repository in /home/mininet/mininet/custom/.git/
mininet@mininet-ofm:~/mininet/custom$ git add scapy_attack.py
mininet@mininet-ofm:~/mininet/custom$ git commit -m 'First commit'
[master (root-commit) b39d1cb] First commit
 1 file changed, 22 insertions(+)
 create mode 100644 scapy_attack.py
mininet@mininet-ofm:~/mininet/custom$ git remote add origin https://github.com/Abhinandini/Lab_7.git
mininet@mininet-ofm:~/mininet/custom$ git push origin master
Username for 'https://github.com': abhinandini.u@gmail.com
Password for 'https://abhinandini.u@gmail.com@github.com':
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 616 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Abhinandini/Lab_7.git
 * [new branch]      master -> master
mininet@mininet-ofm:~/mininet/custom$
```

```
sdn@sdn-controllers:~$ git pull https://github.com/Abhinandini/Lab_7.git
Username for 'https://github.com': abhinandini.u@gmail.com
Password for 'https://abhinandini.u@gmail.com@github.com':
From https://github.com/Abhinandini/Lab_7
 * branch            HEAD       -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 scapy_attack.py | 22 ++++++++++++++++++++++
 1 file changed, 22 insertions(+)
 create mode 100644 scapy_attack.py
sdn@sdn-controllers:~$
sdn@sdn-controllers:~$ git add scapy_detect.py
sdn@sdn-controllers:~$ git commit -m "Add existing file"
On branch master
Untracked files:
        .Xauthority
        .bash_history
        .bash_logout
        .bashrc
        .cache/
        .config/
        .gitconfig
        .karaf/
        .local/
        .oracle_jre_usage/
        .profile
        .python_history
        .sudo_as_admin_successful
        .viminfo
        .wget-hsts
        floodlight/
        karaf-0.8.3.tar.gz
        karaf-0.8.3/
        pyshark_test.py
        ryu/

nothing added to commit but untracked files present
sdn@sdn-controllers:~$ git push -u origin master
Username for 'https://github.com': abhinandini.u@gmail.com
Password for 'https://abhinandini.u@gmail.com@github.com':
Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 795 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/Abhinandini/Lab_7.git
   b39d1cb..d547736  master -> master
Branch master set up to track remote branch master from origin.
sdn@sdn-controllers:~$
```



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 \
   /etc/openvswitch/cacert.pem \
   /var/lib/openvswitch/pki/controllerca/cacert.pem \
2. Provide screenshot of results. [**5 points**]

```
mininet@mininet-ofm:/etc/openvswitch$ sudo ovs-pki req+sign ctl controller
ctl-req.pem     Fri Mar 19 22:50:55 PDT 2021
        fingerprint cfe03f663fba081360997e62c86a4e9c3daea7c6
mininet@mininet-ofm:/etc/openvswitch$
mininet@mininet-ofm:/etc/openvswitch$ sudo ovs-pki req+sign sc switch
sc-req.pem      Fri Mar 19 22:51:01 PDT 2021
        fingerprint fbc621b25b1e36310c65c95ec57615401ae420db
mininet@mininet-ofm:/etc/openvswitch$
mininet@mininet-ofm:/etc/openvswitch$ sudo ovs-vsctl set-ssl \
> /etc/openvswitch/sc-privkey.pem \
> /etc/openvswitch/sc-cert.pem \
> /etc/openvswitch/cacert.pem \
>
mininet@mininet-ofm:/etc/openvswitch$
```

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

.pem are container file format that stores cryptographic keys, SSL certificates and associated private keys.

cert.pem is end user certificate that encrypts HTTPS.

Privkey.pem contains the RSA key generated with the certificate.

Req.pem requests signature from certification authority.

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**]

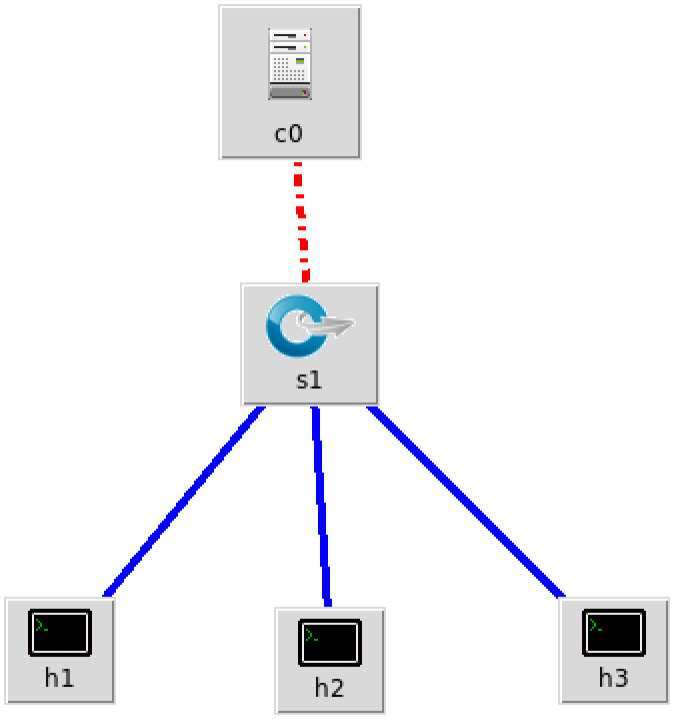
   pssl is passive openflow connection method where listens for SSL on port 6633. The 3 options are PKI configuration necessary to use SSL.

   -p shares the file with private key

   -c shares the certificate for private key

   -C shares the file with peer CA certification.

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**]


c0
s1
h1
h2
h3

mininet@mininet-ofm: ~

```
import cmd
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    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)

    info( '*** Add hosts\n')
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(s1, h3)
    net.addLink(h1, s1)
    net.addLink(h2, s1)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s1').start([c0])

    info( '*** Post configure switches and hosts\n')
    command = net['s1'].cmd('ovs-vsctl show')
    print command
```

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 python script.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h3 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
97060324-da2c-4cc5-ac47-5e2006b159a7
    Manager "ptcp:6632"
    Bridge "s1"
        Controller "ssl:127.0.0.1:6633"
        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"

*** Starting CLI:
mininet>
```

7. Please describe the steps needed to create a SSL secure connection between a switch and a controller in your own words. [**5 points**]

To create SSL secure connection, both switch and controller should support SSL connection. SSL handshake begins. The switch requests the server to share public key and exchanges its own public key. This allows the further communication in encrypted messages. Any message sent from switch will be encrypted using the key send by controller and the controller decrypts using its private key and viceversa.

8. Can you describe the types of attack this objective can help prevent? [**5 points**]

   Man in the middle attack can be prevented as both public and private keys are required to decrypt and read the data exchanged.

Objective 3 – SSL/TLS on Floodlight

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

```
sdn@sdn-controllers:~/floodlight/ssl$ keytool -genkey -keyalg RSA -alias floodlight -keystore keystore.jks -storepass changeit -validity 360 -
keysize 2048
What is your first and last name?
  [Unknown]:  Abhinandini Umesh
What is the name of your organizational unit?
  [Unknown]:  CUB
What is the name of your organization?
  [Unknown]:  CUB
What is the name of your City or Locality?
  [Unknown]:  Boulder
What is the name of your State or Province?
  [Unknown]:  CO
What is the two-letter country code for this unit?
  [Unknown]:  CO
Is CN=Abhinandini Umesh, OU=CUB, O=CUB, L=Boulder, ST=CO, C=CO correct?
  [no]:  yes

Enter key password for <floodlight>
        (RETURN if same as keystore password):

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -import
keystore -srckeystore keystore.jks -destkeystore keystore.jks -deststoretype pkcs12".
sdn@sdn-controllers:~/floodlight/ssl$
sdn@sdn-controllers:~/floodlight/ssl$ keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.p12 -srcstoretype jks -deststor
etype 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
sdn@sdn-controllers:~/floodlight/ssl$
```

```
sdn@sdn-controllers:~/floodlight/ssl$ 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/ssl$
sdn@sdn-controllers:~/floodlight/ssl$ ls
keystore.jks  keystore.p12  keystore.pem
sdn@sdn-controllers:~/floodlight/ssl$ cat keystore.pem
Bag Attributes
    friendlyName: floodlight
    localKeyID: 54 69 6D 65 20 31 36 31 36 33 37 32 32 31 30 39 37 38
Key Attributes: <No Attributes>
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFDjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIhMZkQjJ6vY0CAggA
MBQGCCqGSIb3DQMHBAgmo0yx5yotsQSCBMiIPpR+vaj71Vm6A/ckIwCx5VFQRpNK
XLvXqu6ymZj+vIv6D2vYEVWmhihSMyEqJMdqQY3x6BnYrEvhmMZjX1dc/39Ns0kE
0M9kFC3md25HknfCUunZ8RG89Z2KJ6YVpWlI1P+f2/Y1LJhJq6Nt+iHiRDL6KQBe
gR19dkcgRZMAb2aVm25UDekdwDBBjOKNcJLA/9iMUsfa6hCFsSIfrA5wIYmLiWBq
+xY5bhu9hWVb6X8A1R4O9EaoLM2IaZ6HchqH68Fuja3226Fq+P2UPtHqxvM4J2fQ
rcuVkKHVMbgaAD7auGywOm0oClatDSZncCFwP/SbygcnqvEbaJOW8/+e03fBaMbM
dYE2ohE4UllSyJuK8Go+rG63wW+9wGM62yeWpwwsAIhyeNpTSxW98GScqSHOrCZ0
aDJqIs+zQpuoXgMIZhXm6yuweLZz5jTkSEynUe6D2WxpbUvBs3To2OGWbd7tyDhG
voHN2TiCmajlKFsGveYqVsR839XKN4f6fzPEbQX8x+DklJrxM3AuOwRzd7x8/zDI
3RHpruXu6YSPCU6ToQgNn7LPySd6JO21BJspTpkTiahD5xdVH+sOr665bk8yzBH2
arXHtdEfH8I03cw+JBnYSmeios7XmiSSkQvHEV5pGBo0o+UPAfrbQ0qFaXDPn4ni
FcECy/tavGyDVAmPH51Q3sVV35gUx8WclV4XEN69OFNkkTUZuhodWs9qqVFbcZdq
RNgonzE3KFyeRqvCkLE1XaIW+RKFcrXKkdgxw9YRHiV1MRmuGHZnDHzJ4V0iuMhV
EShUBAmLBKCmMBcYWcqXxvwYF1zV4FLax0BXRTjc/MllJ7T7iYYspjXWdvkP6GE0
EsFqkf9K4kAwfMBMQYg0aRDF1DRsncTLCCbPsxiUrmnNzlehTcvECYLLx6rbbWAn
J3CCENnb7+TOl6CHX48wIfqa4j0lEj3M+D6ep/TAuZB9ePuLKxM2yWPGEg8HHIyH
mAyk+ebRx9cYZKxTVB5uwu52v19MggUIfmkIqS9EJVXc+BW7UxiVBjt/kCV0DBkL
5A0cL1DCwprF/P+KrNMCue6yOGaQtwAK7cn+8X99U9vRIst1slEOtt8KubURfV91
NAWRmkpGbunOwzv7McxEXHbcKkFaSqIEFeUHDKl2P8O4Jt7UEZ6vlx4hHz7wf9iB
0Wjq5jDfaHoS8ALEPj1+IKSzMz7EKLPfWorOy2wMivf6MNjpG9bjvlGdXG5B/nl0
NNxHLQ/4/PcTKDNlF9vfZxrTfvZYxECykj50SVKUy+oHVAuN6S/N0tCI871I9XB7
```

```
    friendlyName: floodlight
    localKeyID: 54 69 6D 65 20 31 36 31 36 33 37 32 32 31 30 39 37 38
subject=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
issuer=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
-----BEGIN CERTIFICATE-----
MIIDZzCCAk+gAwIBAgIEc3xfAjANBgkqhkiG9w0BAQsFADBkMQswCQYDVQQGEwJD
TzELMAkGA1UECBMCQ08xEDAOBgNVBAcTB0JvdWxkZXIxDDAKBgNVBAoTA0NVQjEM
MAoGA1UECxMDQ1VCMRowGAYDVQQDExFBYmhpbmFuZGluaSBVbWVzaDAeFw0yMTAz
MjIwMDE2MjlaFw0yMjAzMTcwMDE2MjlaMGQxCzAJBgNVBAYTAkNPMQswCQYDVQQI
EwJDTzEQMA4GA1UEBxMHQm91bGRlcjEMMAoGA1UEChMDQ1VCMQwwCgYDVQQLEwND
VUIxGjAYBgNVBAMTEUFiaGluYW5kaW5pIFVtZXNoMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAiiNASU0daH8YR34zm0roQ9adkoMzHI0cmHs22GpXJ6/s
5BaHugNOPmcsvsS4hZq8njz4U3RLy7P7RAPeBtF/J/sHWjSdc84IcP+KaBIq1Dxr
opdfs4y6bXxlhEJdjfM5ER4L5K3+qDy0eNyOrvjjHdqj1dMBHT6zO+u0max41idw
3PekZH1eazTiszUjWPH9yoMXzVkyKNCGkrT76rX0vhAJGNi8d7tOGfqo/yGNH5eE
R7iAPf6df8DLIN/CY1x+HmMl1iN51zFSpuyUqA/SM8oXrU9c1Ia9/VnGdxPkpjZW
gALWo1LW5sEYPMj+XxJvlVFGBhkm60I0ysejPraQUwIDAQABoyEwHzAdBgNVHQ4E
FgQUkQiXOQqjQwe7m5VAx9sefAZKKtAwDQYJKoZIhvcNAQELBQADggEBAHovCMgT
jAzReI/9M0CAU6GllPXS3zsJPrpGpF4iimAifzJh8g4eP0CviKmJIp707eF/+KCf
gOQJSASgU5c8fQnIEwJxgPc8KKW1+qghfVbSR0kVLekoEF3paa3vgqMPhITgKlB8
g+aso89zS1AB4pgAvg/jnHv3SpCIfYLK9y3wV2E/WR1jR0X69XroTM1HVv2EG6Q8
Mc/KMrx9ZanJTWae7TV9Ipwxdm40SDFJegkgrE3pXdC341YRjrVzAsf0EIpmMz0m
DfEtt9YWJ6U9m9kolw1+mgABXCzoBIzOLF/sSbuP8UgCrnjo2WmvB41NwR35Rial
va9H2Uw9D/U0Mx0=
-----END CERTIFICATE-----
sdn@sdn-controllers:~/floodlight/ssl$
sdn@sdn-controllers:~/floodlight/ssl$ vi cacert.pem
sdn@sdn-controllers:~/floodlight/ssl$ cat cacert.pem
Bag Attributes
    friendlyName: floodlight
    localKeyID: 54 69 6D 65 20 31 36 31 36 33 37 32 32 31 30 39 37 38
subject=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
issuer=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
-----BEGIN CERTIFICATE-----
MIIDZzCCAk+gAwIBAgIEc3xfAjANBgkqhkiG9w0BAQsFADBkMQswCQYDVQQGEwJD
TzELMAkGA1UECBMCQ08xEDAOBgNVBAcTB0JvdWxkZXIxDDAKBgNVBAoTA0NVQjEM
```

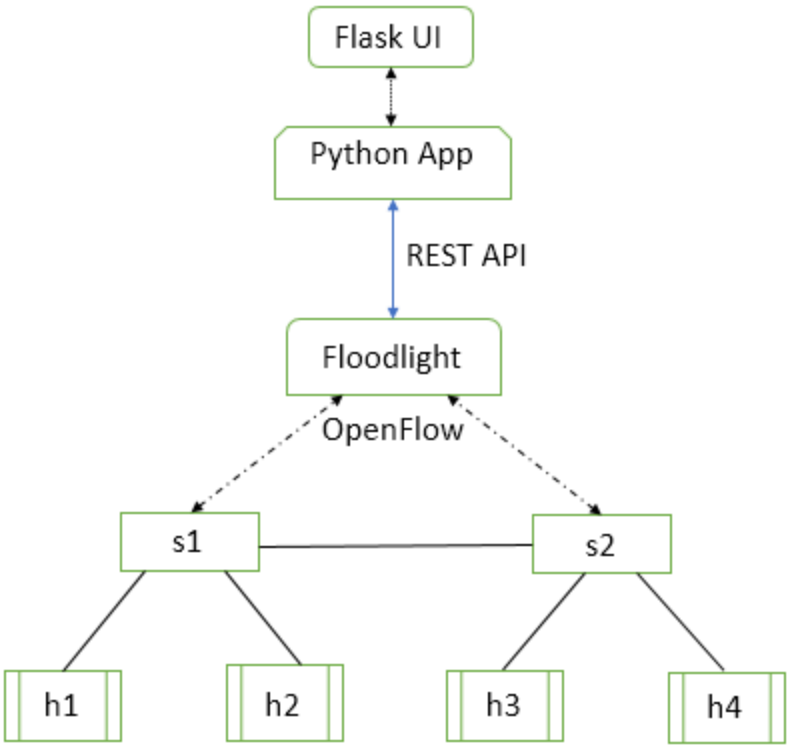
```
sdn@sdn-controllers:~/floodlight/ssl$ cat cacert.pem
Bag Attributes
    friendlyName: floodlight
    localKeyID: 54 69 6D 65 20 31 36 31 36 33 37 32 32 31 30 39 37 38
subject=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
issuer=/C=CO/ST=CO/L=Boulder/O=CUB/OU=CUB/CN=Abhinandini Umesh
-----BEGIN CERTIFICATE-----
MIIDZzCCAk+gAwIBAgIEc3xfAjANBgkqhkiG9w0BAQsFADBkMQswCQYDVQQGEwJD
TzELMAkGA1UECBMCQ08xEDAOBgNVBAcTB0JvdWxkZXIxDDAKBgNVBAoTA0NVQjEM
MAoGA1UECxMDQ1VCMRowGAYDVQQDExFBYmhpbmFuZGluaSBVbWVzaDAeFw0yMTAz
MjIwMDE2MjlaFw0yMjAzMTcwMDE2MjlaMGQxCzAJBgNVBAYTAkNPMQswCQYDVQQI
EwJDTzEQMA4GA1UEBxMHQm91bGRlcjEMMAoGA1UEChMDQ1VCMQwwCgYDVQQLEwND
VUIxGjAYBgNVBAMTEUFiaGluYW5kaW5pIFVtZXNoMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAiiNASU0daH8YR34zm0roQ9adkoMzHI0cmHs22GpXJ6/s
5BaHugNOPmcsvsS4hZq8njz4U3RLy7P7RAPeBtF/J/sHWjSdc84IcP+KaBIq1Dxr
opdfs4y6bXxlhEJdjfM5ER4L5K3+qDy0eNyOrvjjHdqj1dMBHT6zO+u0max41idw
3PekZH1eazTiszUjWPH9yoMXzVkyKNCGkrT76rX0vhAJGNi8d7tOGfqo/yGNH5eE
R7iAPf6df8DLIN/CY1x+HmMl1iN51zFSpuyUqA/SM8oXrU9c1Ia9/VnGdxPkpjZW
gALWo1LW5sEYPMj+XxJvlVFGBhkm60I0ysejPraQUwIDAQABoyEwHzAdBgNVHQ4E
FgQUkQiXOQqjQwe7m5VAx9sefAZKKtAwDQYJKoZIhvcNAQELBQADggEBAHovCMgT
jAzReI/9M0CAU6GllPXS3zsJPrpGpF4iimAifzJh8g4eP0CviKmJIp707eF/+KCf
gOQJSASgU5c8fQnIEwJxgPc8KKW1+qghfVbSR0kVLekoEF3paa3vgqMPhITgKlB8
g+aso89zS1AB4pgAvg/jnHv3SpCIfYLK9y3wV2E/WR1jR0X69XroTM1HVv2EG6Q8
Mc/KMrx9ZanJTWae7TV9Ipwxdm40SDFJegkgrE3pXdC341YRjrVzAsf0EIpmMz0m
DfEtt9YWJ6U9m9kolw1+mgABXCzoBIzOLF/sSbuP8UgCrnjo2WmvB41NwR35Rial
va9H2Uw9D/U0Mx0=
-----END CERTIFICATE-----
sdn@sdn-controllers:~/floodlight/ssl$
sdn@sdn-controllers:~/floodlight/ssl$ rm keystore.pem keystore.p12
sdn@sdn-controllers:~/floodlight/ssl$ pwd
/home/sdn/floodlight/ssl
```

```
sdn@sdn-controllers:~/floodlight$ vi cacert.pem
sdn@sdn-controllers:~/floodlight$ pwd
/home/sdn/floodlight
sdn@sdn-controllers:~/floodlight$ cd ssl/
sdn@sdn-controllers:~/floodlight/ssl$ ls
cacert.pem  keystore.jks
sdn@sdn-controllers:~/floodlight/ssl$ keytool -import -alias "openvswitch" -keystore keystore.jks -file /home/sdn/floodlight/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
```

```
mininet@mininet-ofm:~/mininet/custom$ sudo python ssl_floodlight.py
*** Adding controller
Unable to contact the remote controller at 192.168.56.109:6653
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h2 h1 h3
*** Starting controllers
*** Starting switches
None
*** Post configure switches and hosts
*** Starting CLI:
mininet> sh ovs-vsctl show
97060324-da2c-4cc5-ac47-5e2006b159a7
    Bridge "s1"
        Controller "ssl:192.168.56.109:6653"
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth3"
            Interface "s1-eth3"
    ovs_version: "2.0.2"
mininet>
```

Objective 4 – REST Static flow entries and Firewall via Python

In this objective, you will be writing a Python script that uses Flask 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 will removing this line do? [**5 points**]

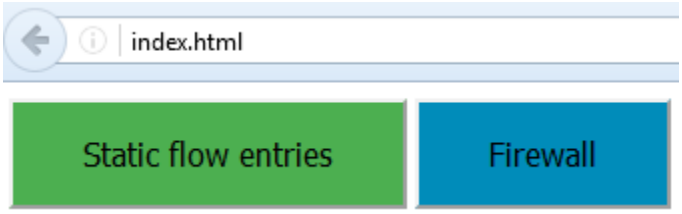
   Forwarding is the default module which inserts flows in switches that routes the incoming packets after the topology is learnt. By deleting, there won't be any flow entries in the switch.

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**]

   No the ping will not work and there is only one flow with action to send to controller. This is because the controller has no ability to learn the forwarding methods.

   The ping is not successful. No Netflow or Switch flow is

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.



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**]
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.
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