**Lab: Automated Defense and Pretense against Distributed Denial of Service Attacks**

*Dr. Prasad Calyam (Contact:* [*calyamp@missouri.edu*](mailto:calyamp@missouri.edu)*)*

Compiled by Ramya Payyavula

1. **Purpose of the Lab**

To create a cloud infrastructure that hosts a web server on GENI cloud testbed, faking a DDoS attack on the virtual machine hosting the service to make the service unavailable. Detecting a DDoS attack and applying pretense methodology on the attacker, resulting in a smooth traffic to the actual virtual machine hosting web server. Understanding how lack of adequate defense strategies can affect a cloud service provider in LoA. Learning various tools like Frenetic, Mininet, Scapy and OvSwitch.

Compared to the previous MTD lab, in this exercise you will learn more about how to setup network policies for monitoring attacks, blacklisting attacker flows, and applying a defense using pretense via Scapy. You will also realize how to analyze network flows at root switch to understand the attack has no impact on the target application.

***Docker container for labs could be used; scripts to automate steps – think where should student spend time? Keep central config file as needed – make it easy for reproducibility.***

1. **References to guide Lab Work**

* **OCaml Package Manager**

<https://opam.ocaml.org/>

* **Creating a LAMP stack**

<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04>

* **Frenetic Network Programming**

<https://github.com/frenetic-lang/manual/blob/master/programmers_guide/frenetic_programmers_guide.pdf>

* **OvS Open vSwitch**

<http://docs.openvswitch.org/en/latest/faq/openflow/>

* **Scapy**

<https://scapy.net/>

* **Critical Reading**

1.<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.2913&rep=rep1&type=pdf>

2. <https://ieeexplore.ieee.org/abstract/document/6883418>

3. <https://www.bostonglobe.com/business/2016/10/21/yes-your-internet-has-probably-been-less-than-perfect-today/kISaYiya1qsNw9kkvEakrJ/story.html>

4. <https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage>

5. <https://www.calyptix.com/top-threats/ddos-attacks-2018-new-records-and-trends/>

6. <https://www.cs.princeton.edu/~jrex/papers/icfp11.pdf>

7. <https://www.digitalocean.com/community/tutorials/how-to-migrate-a-mysql-database-to-a-new-server-on-ubuntu-14-04>

1. **Lab steps and output collection guidelines**

To complete this Lab, you will need to perform the following steps:

In the first step, you will setup the testbed required for the experiment, then install the controller, root switch and slave switch. After this, you will install a LAMP stack, setup controller, root switch, slave switch and frenetic. You will also setup default routing between the root switch and slave switch and capture the packets flowing between the machines. Then, using network Policies, detect an attacker and move the attacker flow to the quarantine machine to deceive the attacker. You can observe the network flow and all the other information regarding the server, users, devices, attacker and quarantine machines on the public routable IP of the controller.

* 1. **Setup a slice with a controller, root-switch, slave-switch, quarantine machine, user and an attacker.**

1. Login to GENI portal and create a slice or load an existing empty slice for the lab. We are going to reserve the resources using Graphical Interface of the portal (Jacks). Reserve your controller device in a separate slice by itself. That way if there is a failure on another aggregate, you can delete that slice and try again without losing all your setup on the controller.

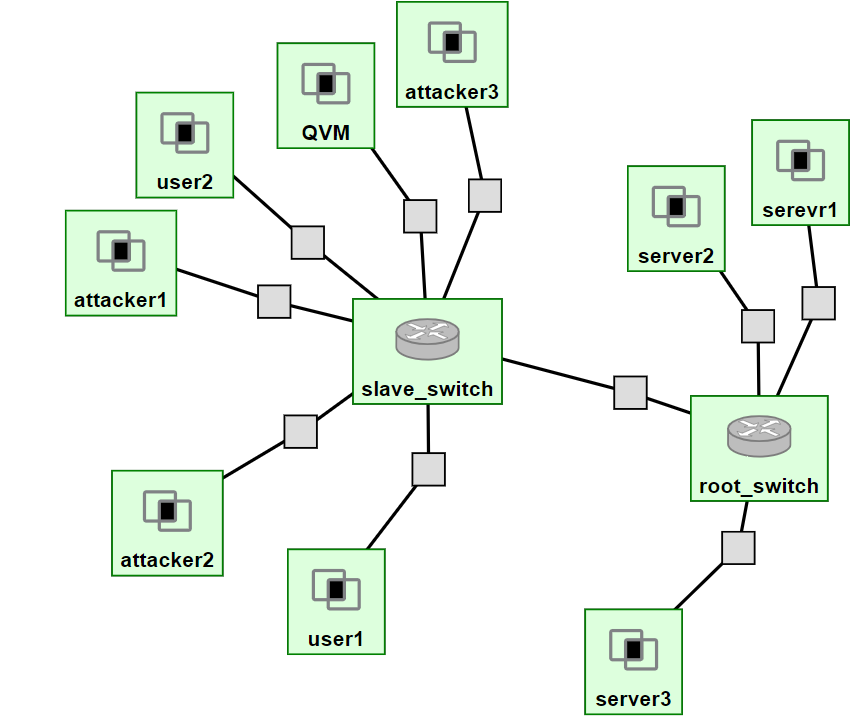
### Load a simple topology in Jacks

|  |
| --- |
| 1. In the Portal, open the Slice page for the slice you just created (Click on slices, select project and add name to the slice). Notice that you can create the slice with omni or Jacks. 2. Press the **Add Resources** button to launch Jacks for this slice. 3. From the **Choose Rspec** menu, select ‘Files’ option to load the provided Rspec file for Controller and other aggregates in respective slices. 4. Once it is done, a few VMs, a root-switch, a slave-switch, users and attackers appear in one slice. And a controller appears in the controller slice. 5. Select the site name as cornellInstageni or clemsonInstageni. 6. Choose an aggregate to select where to reserve the resources and then click on “Reserve resources”. |

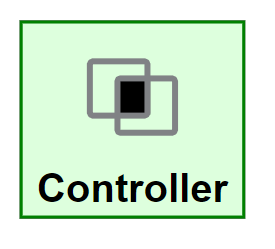
The topology you loaded has three VM types and a switch:

* 1. Server
  2. Controller
  3. Attacker
  4. User

The sliver should look something like this:



The controller sliver should look something like this:



* 1. **Controller Installation**

The Controller is used to capture the network traffic and packet flow on root-switch and slave-switch, calculate the suspicious score and detect an attacker. So, we should Install LAMP Stack on Controller. Give you mysql database password as root during installations.

* Update Ubuntu and all installed Software’s

|  |
| --- |
| *sudo apt-get update* |

* Install LAMP, Python, Opam, Mininet. **Give mysql password as “root”**

|  |
| --- |
| *sudo apt-get install apache2 python-setuptools python-dev build-essential libcurl4-openssl-dev opam curl apt dpkg mininet m4 zlib1g-dev python-pip libmysqlclient-dev apache2 php libapache2-mod-php libssl-dev* |

* Initialize Opam If environment variables or not setup run. Add the opam environmental config to the frenetic user's profile so it runs each time a shell is opened. Check the environment variables again.

|  |
| --- |
| *sudo opam init -y*  *sudo opam switch 4.06.0*  *sudo opam switch*  *eval `opam config env`*  *sudo echo 'eval `opam config env`' >>.profile*  *ifconfig* |

*#NOTE the details of the public facing ethernet port and public facing ip address for*

*future reference.*

* Install Frenetic and Initial startup of frenetic with no switches connected

|  |
| --- |
| *sudo opam pin add frenetic -y* [*https://github.com/frenetic-*](https://github.com/frenetic-)*lang/frenetic.git*  *pip install pycurl frenetic mysql*  *sudo opam install frenetic*  *~/.opam/4.06.0/bin/frenetic http-controller --verbosity debug* |

* Create MySQL database

|  |
| --- |
| *sudo apt-get -y install mysql-server apache2 php php-mysql libapache2-mod-php*  *sudo apt-get -y install php-curl* |

Type in password “root “and note it for future use.

|  |
| --- |
| cd /etc/mysql/mysql.conf.d  sudo sed -i "s/.\*bind-address.\*/bind-address = 0.0.0.0/g" mysqld.cnf  mysql -u root -p **#enter your mysql password when prompted.**  create database test;  CREATE USER 'monty'@'%' IDENTIFIED BY 'some\_pass';  GRANT ALL PRIVILEGES ON \*.\* TO 'monty'@'%';  CTRL+D; |

|  |
| --- |
| *sudo ufw allow 3306;*  *sudo service mysql restart;* |

***Note:*** *If you would like to login to the database in any of the switches use following commands*

*mysql -u monty -h <controllerIP> -p*

*Example: mysql -u monty -h '72.36.65.106' -p*

*enter password as 'some\_pass'*

* 1. **Root-switch Installation**

Each root-switch installation may be slightly different depending on how many devices are connected to each.

* Update Ubuntu and all installed Software’s

|  |
| --- |
| *sudo apt-get update* |

* Install Tshark to capture OpenFlow packets (select yes when prompted)

|  |
| --- |
| *sudo apt-get install -y openvswitch-switch tshark* |

* Create a bridge br0

|  |
| --- |
| *sudo ovs-vsctl add-br br0* |

* Activate network interface connected to the root switch and add network interface ports to the bridge0

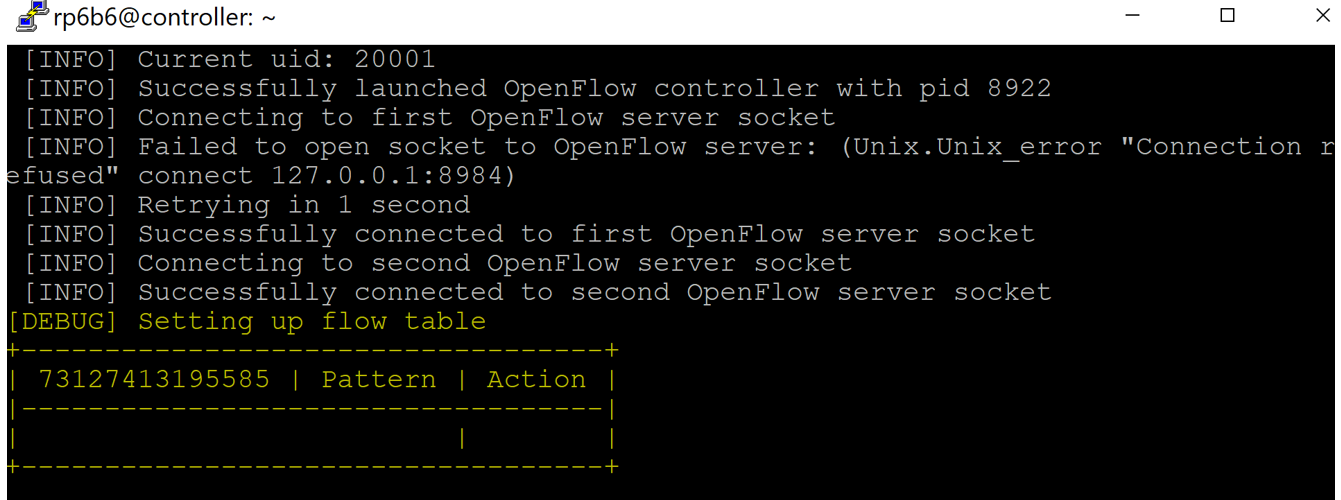
|  |
| --- |
| *sudo ifconfig eth1 0*  *sudo ifconfig eth2 0*  *sudo ifconfig eth3 0*  *sudo ifconfig eth4 0*  *sudo ovs-vsctl add-port br0 eth1*  *sudo ovs-vsctl add-port br0 eth2*  *sudo ovs-vsctl add-port br0 eth3*  *sudo ovs-vsctl add-port br0 eth4* |

* Configure Controller on bridge br0**, replace your controller IP in IP Address of the Controller**

|  |
| --- |
| *sudo ovs-vsctl set-controller br0 tcp:<IP Address of the Controller>:6633* |

**Example:** sudo ovs-vsctl set-controller br0 tcp:72.36.65.106:6633

* With frenetic running, view the current switches and note the DPID number displayed for the switch. This is a 14+ digit number used to identify the switch later.



* 1. **Slave-switch Installation**

Each slave-switch installation may be slightly different depending on how many devices are connected to each.

* Update Ubuntu and all installed Software’s

|  |
| --- |
| *sudo apt-get update* |

* Install Tshark to capture OpenFlow packets

|  |
| --- |
| *sudo apt-get install -y openvswitch-switch tshark* |

* Create a bridge br0

|  |
| --- |
| *sudo ovs-vsctl add-br br0* |

* Activate network interface connected to the slave switch and add network interface ports to the bridge0

|  |
| --- |
| *sudo ifconfig eth1 0*  *sudo ifconfig eth2 0*  *sudo ifconfig eth3 0*  *sudo ifconfig eth4 0*  *sudo ifconfig eth5 0*  *sudo ifconfig eth6 0*  *sudo ifconfig eth7 0*  *sudo ovs-vsctl add-port br0 eth1*  *sudo ovs-vsctl add-port br0 eth2*  *sudo ovs-vsctl add-port br0 eth3*  *sudo ovs-vsctl add-port br0 eth4*  *sudo ovs-vsctl add-port br0 eth5*  *sudo ovs-vsctl add-port br0 eth6*  *sudo ovs-vsctl add-port br0 eth7* |

* Configure Controller on bridge br0, **replace your controller IP in IP Address of the Controller**

|  |
| --- |
| *sudo ovs-vsctl set-controller br0 tcp:<IP Address of the Controller>:6633* |

**Example:** sudo ovs-vsctl set-controller br0 tcp:128.206.119.34:6633

* With frenetic running, view the current switches and note the DPID number displayed for the switch. This is a 14+ digit number used to identify the switch later.



* 1. **Installing AdminUI on the controller**

AdminUI monitors the cloud network, keeps track of network components and network traffic. Policies generated by Frenetic can be updated using JSON arrays. Automatically blacklists the attacker once an attack is detected and updates the policies.

* set the apache default site to point to /var/www/public\_html at DocumentRoot and restart apache2 server

|  |
| --- |
| *cd /etc/apache2/sites-enabled*  *sudo sed -I “s+.\*DocumentRoot.\*+DocumentRoot /var/www/public\_html/Dolus\_DDos/public+g” 000-default.conf*  *sudo sed -i s+var/www/html+var/www/public\_html+g 000-default.conf*  *cd /var/www*  *sudo mv /var/www/html /var/www/public\_html*  *cd /var/www/public\_html*  *sudo service apache2 restart* |

Open your controller IP in the browser and see the apache page.

* Run the following Scripts to clone the Dolus\_DDos repository

|  |
| --- |
| *cd /var/www/public\_html*  *sudo rm index.html*  *sudo echo "<?php echo phpinfo(); ?>" > index.php*  *cd*  *sudo apt-get -y install composer*  *cd /var/www/public\_html*  *sudo git clone https://github.com/RamyaPayyavula/Dolus\_DDos*  *cd Dolus\_DDos*  *sudo chmod -R 777 /var/www/*  *sudo chmod -R 774 /var/www/public\_html/Dolus\_DDos* |

* Install Laravel and composer in user frenetic

|  |
| --- |
| *sudo apt-get -y install php-mbstring*  *sudo apt install php-xml*  *composer update*  *composer dump-autoload*  *sudo apt-get install php7.0-zip*  *composer global require "laravel/installer"*  *sudo a2enmod rewrite*  *cd /etc/apache2/sites-enabled*  *sudo vim 000-default.conf* |

**Open 000-default.conf, change /*var/www/public\_html* to */var/www/public\_html/Dolus\_DDos/public* and Add the following code directory after the DocumentRoot line.**

|  |
| --- |
| *<Directory /var/www/public\_html/Dolus\_DDos/public>*  *Options Indexes FollowSymLinks*  *AllowOverride All*  *Require all granted*  *</Directory>* |

Open php.ini file and at line 894, uncomment the line by removing the semicolon before : extension=php\_pdo\_mysql.dll

|  |
| --- |
| *sudo vim /etc/php/7.0/apache2/php.ini* |

After this, execute the following commands to finish the apache installation, configure Dolus, and install Frenetic:

|  |
| --- |
| *cd /var/www/public\_html*  *sudo service apache2 restart*  *cd /var/www/public\_html/Dolus\_DDos*  *composer install*  *sudo chmod -R 755 /var/www/public\_html/Dolus\_DDos*  *sudo chmod -R 777 storage*  *sudo chmod -R 775 bootstrap/cache/*  *sudo chmod -R 775 .env*  *php artisan key:generate*  *sudo pip install sqlalchemy*  *sudo pip install pymysql frenetic pycurl* |

* Change the Database details in settings.py and database. php and restart the apache server. **If your database password is root, you don’t have to do this step.**

Execute the following commands in the Controller Terminal.

**edit database name, username, and password in following files.**

|  |
| --- |
| *sudo vim app/Python/settings.py*  *sudo vim config/database.php*  *sudo vim .env* |

* Create all the tables in the database by executing the models.py script.

|  |
| --- |
| *python app/Python/models.py* |

**Don’t run the following command twice as most of the tables in the database have primary keys, running the command second time will throw an integration error. If you would like to run it twice please delete the data from database and run it again.**

* Changed your slave switch DPID and root switch DPID’s in place of the switchID.

|  |
| --- |
| *sudo vim app/Python/*[*defaultDataInsertion.py*](https://github.com/RamyaPayyavula/Dolus_DDos/blob/master/app/Python/defaultDataInsertion.py) |



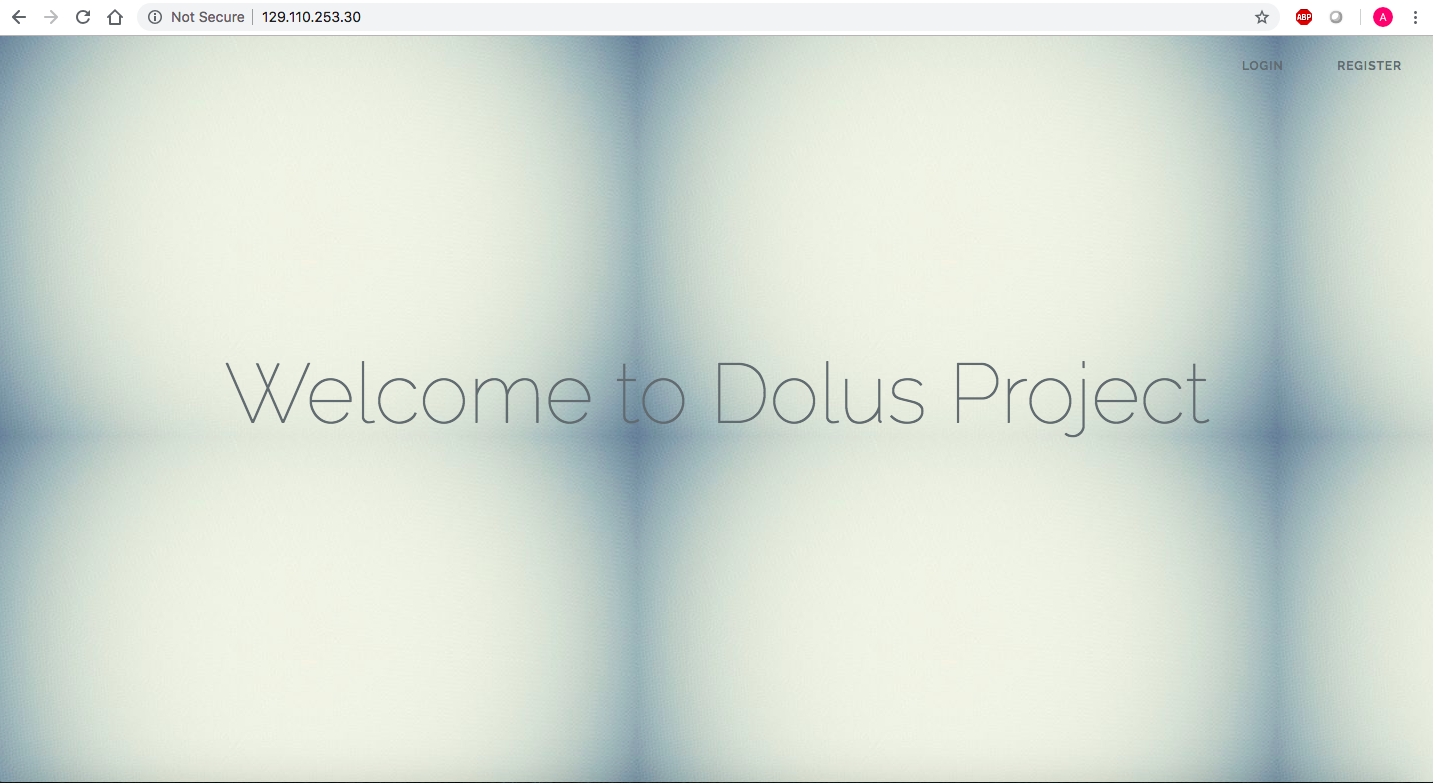
Slave switch DPID

Root switchDPID

|  |
| --- |
| *python app/Python/*[*defaultDataInsertion.py*](https://github.com/RamyaPayyavula/Dolus_DDos/blob/master/app/Python/defaultDataInsertion.py) |

Check the whether the tables created in database or not in second Terminal.

**Open your controller Ip in the browser and register for the AdminUI using your name and log into It. Ensure that the password you set for the AdminUI is 6 characters long.**



**Don’t panic some graphs will be empty as there is no network data flows happened till now.**

* 1. **Run Frenetic and setup default routing**

First, log in as the frenetic user and start up frenetic on the controller. If frenetic is already running, hit Ctrl+c to stop the process. Then, start up frenetic by executing the following commands:

|  |
| --- |
| *eval `opam config env` frenetic http-controller --verbosity debug* |

Once frenetic has been started, blank OpenFlow tables should be displayed in the console.

Since routing has not been configured yet, we’re going to take care of it now. To do so, log into the root-switch and run the ifconfig command. You will notice that there are 5 ethernet interfaces and one loopback interface. To configure the routing, we’ll need to match the MAC address to the settings contained in the openNetwork-ADAPTS.py file on the controller. Connect to the controller via SSH and execute the command cd /var/www/public\_html/Dolus\_DDos/app/Python/ to navigate to the directory containing the file we’re going to edit.

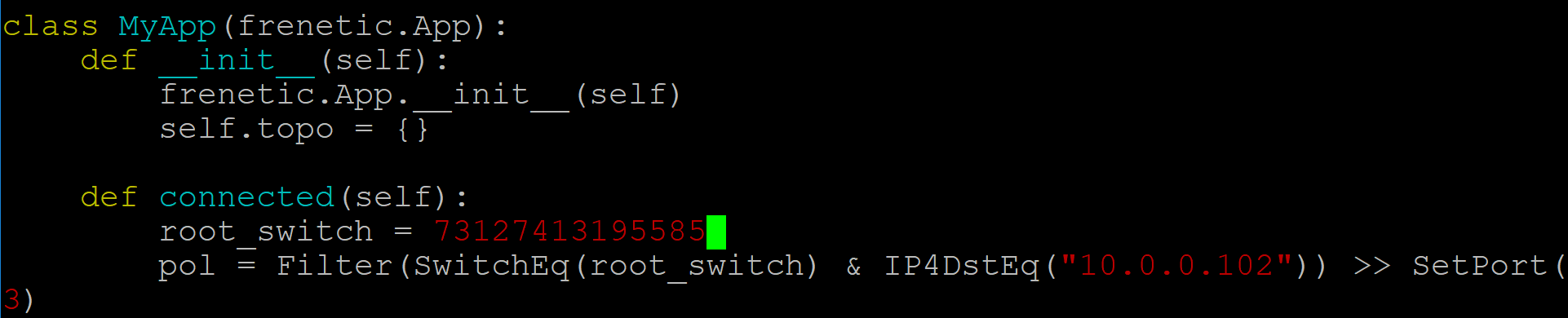
Now that we’re in the directory, execute the command nano openNetwork-ADAPTS.py

to open the file (You’re welcome to use vim if you feel more comfortable with it,

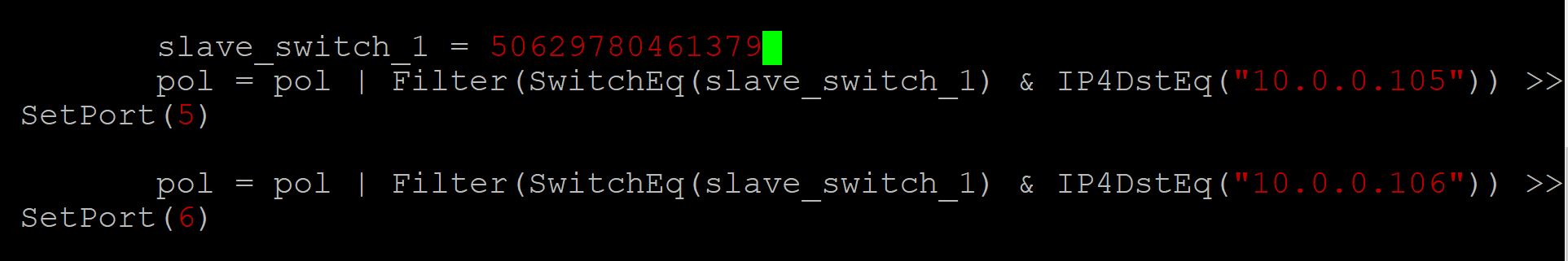
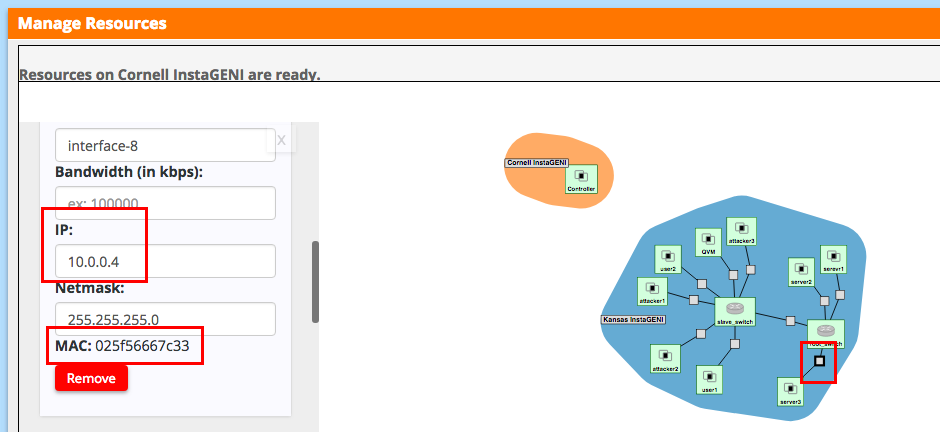
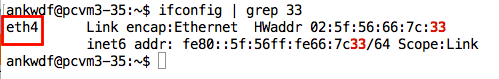
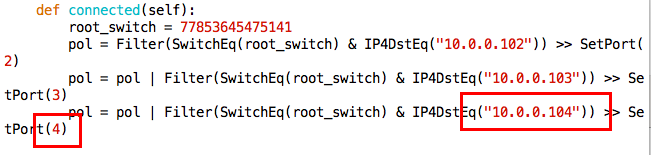
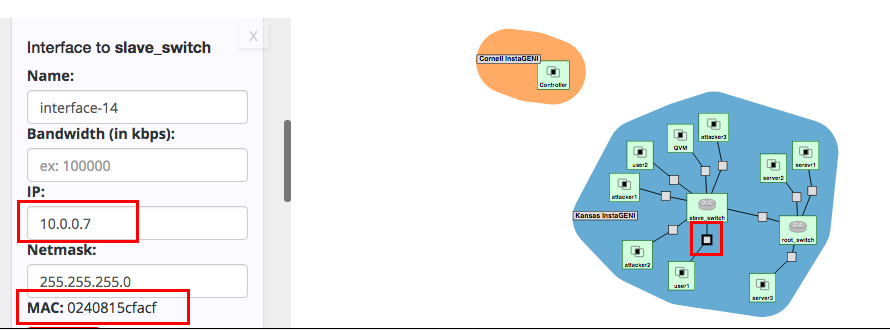
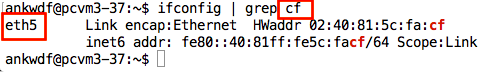
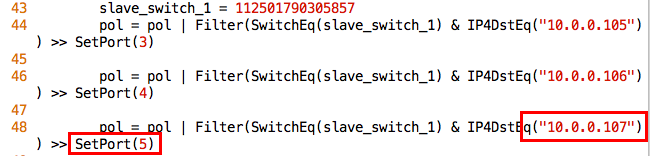
but I chose nano because the commands are at the bottom of the screen in the text editor).

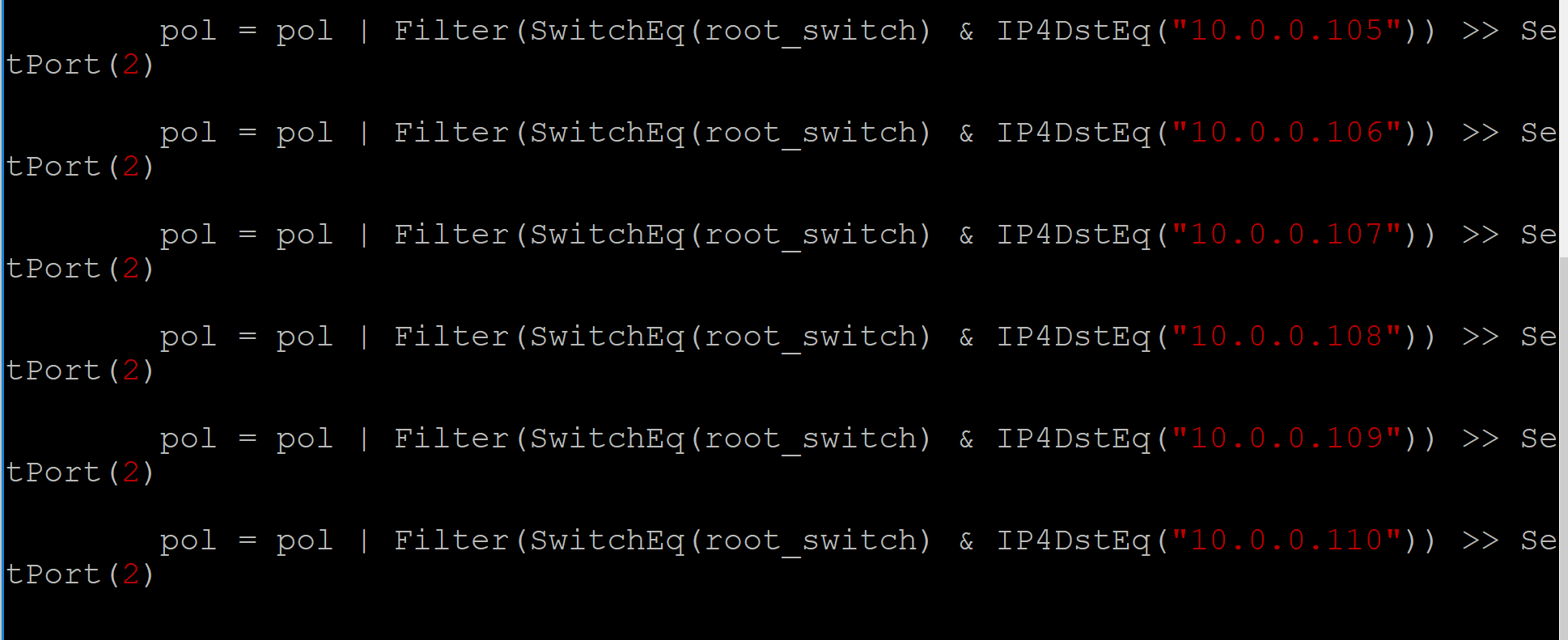
Using the arrow keys, navigate down to the first instance of root\_switch. It should be on line 23, and the DPID number is being assigned to it. Change the DPID number in there to the DPID number of the root-switch that you recorded earlier, then find the slave\_switch\_1 variable. It should be on line 43, and also has a DPID number assigned to it. Change this DPID number to the slave-switch's DPID number as well. After changing the DPID numbers, we’re going to want to change the port numbers specified in the file. To do this, we’ll make a note of each connection’s IP and MAC address, then go on the switch and grep for it, finding the corresponding interface number. Once we do that, we’ll change the interface number for each server in the openNetwork-ADAPTS.py file to the one that matches its hardware ID. This portion is a bit complicated, but using the steps below, you will be able to correctly configure the system:

1. Change you root switch DPID in openNetwork-ADAPTS.py file like below

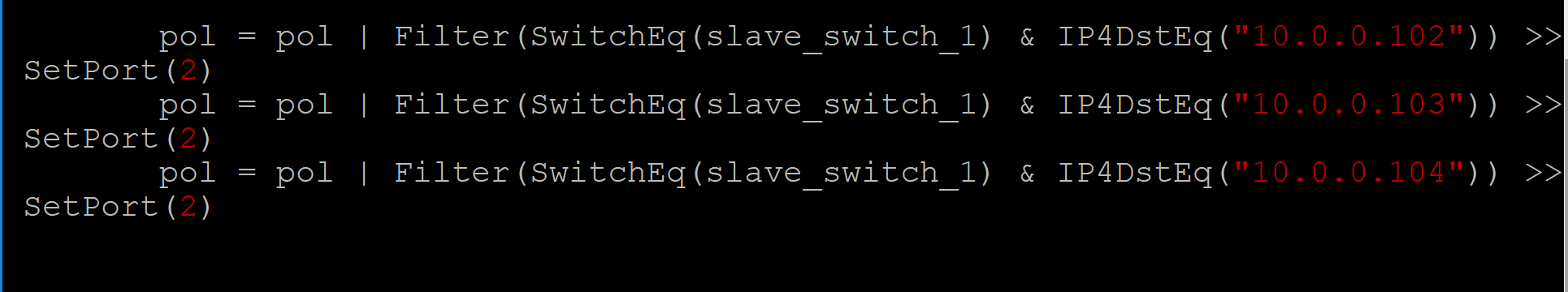


z

1. And slave switch DPID in openNetwork-ADAPTS.py file like below 
2. Then, go to the GENI interface and click on the link between the root-switch and server3 as shown in the image below on the right side with a red box around it. This will open a tab on the left that will display the IP address and the MAC address of the interfaces connecting the root-switch and server3. Scroll down to the interface labeled Interface to root switch and make a note of the IP address and MAC address. For your convenience, the image below on the left-hand side shows an example of where they are located on the page:  
   
3. Then go back into the root-switch console. Type the command ifconfig | grep $num, replacing $num with the last two characters of the MAC address. It should return the interface number as shown below:  
   
4. Please note that multiple MAC addresses may share the last two characters; If multiple interfaces are returned, select the one that completely matches the MAC address you recorded.
5. Now that you have the interface number, go back into the controller and open the openNetwork-ADAPTS.py file if it is not open already. Navigate to the root\_switch variable on line 23 again, and below it will be the port designations based on the IP address. What we’re going to want to do is change the number inside SetPort() to the number of the network interface on the line that matches the IP address that is tied to the MAC address you recorded previously. As you can see in the following picture, I changed the number inside SetPort() to 4 on the line that had the IP address 10.0.0.4:  
   
6. Repeat steps 1 through 3 for server2 and server1 as well. Once this has been completed, log into the slave-switch. The steps 1 through 3 above are essentially the same as what we’ll do for this switch, but I’ve included the steps below so that visual examples can be illustrated for the slave-switch portion as well.
7. Go on to the GENI interface again and click the link between user1 and slave-switch. The sidebar will open containing the IP address and MAC address information. Please go down to the Interface to slave-switch section and make a note of the IP address and MAC address. The image below shows where they are located on the page:  
   
8. After you obtain the IP address and MAC address, go into the slave-switch console and type the command ifconfig | grep $num, replacing $num with the last two characters of the MAC address. It should return the interface number as shown below:  
   
9. Once you obtain the interface number, log back onto the controller and open the openNetwork-ADAPTS.py file if it is not open already. Navigate to the slave\_switch\_1 variable, and below it will be the port configurations for each server connected to it. Find the line containing the IP address you recorded previously and replace the number inside the SetPort() function to the interface number you obtained from the slave-switch console. In the below example, the IP address was 10.0.0.7 and the MAC address of the server that has this IP address matched the interface eth5. Therefore, the number inside the SetPort() function needed to be changed to 5 as shown below:  
   
10. Repeat steps 5 through 7 for each server connected to the slave-switch, and then save the openNetwork-ADAPTS.py file on the controller.
11. In root switch for all VM’s connected to slave switch, SetPort should be set to the root switch port number (check the eth numbers like you did before for other VM’s) to direct the traffic’s to slave switch.



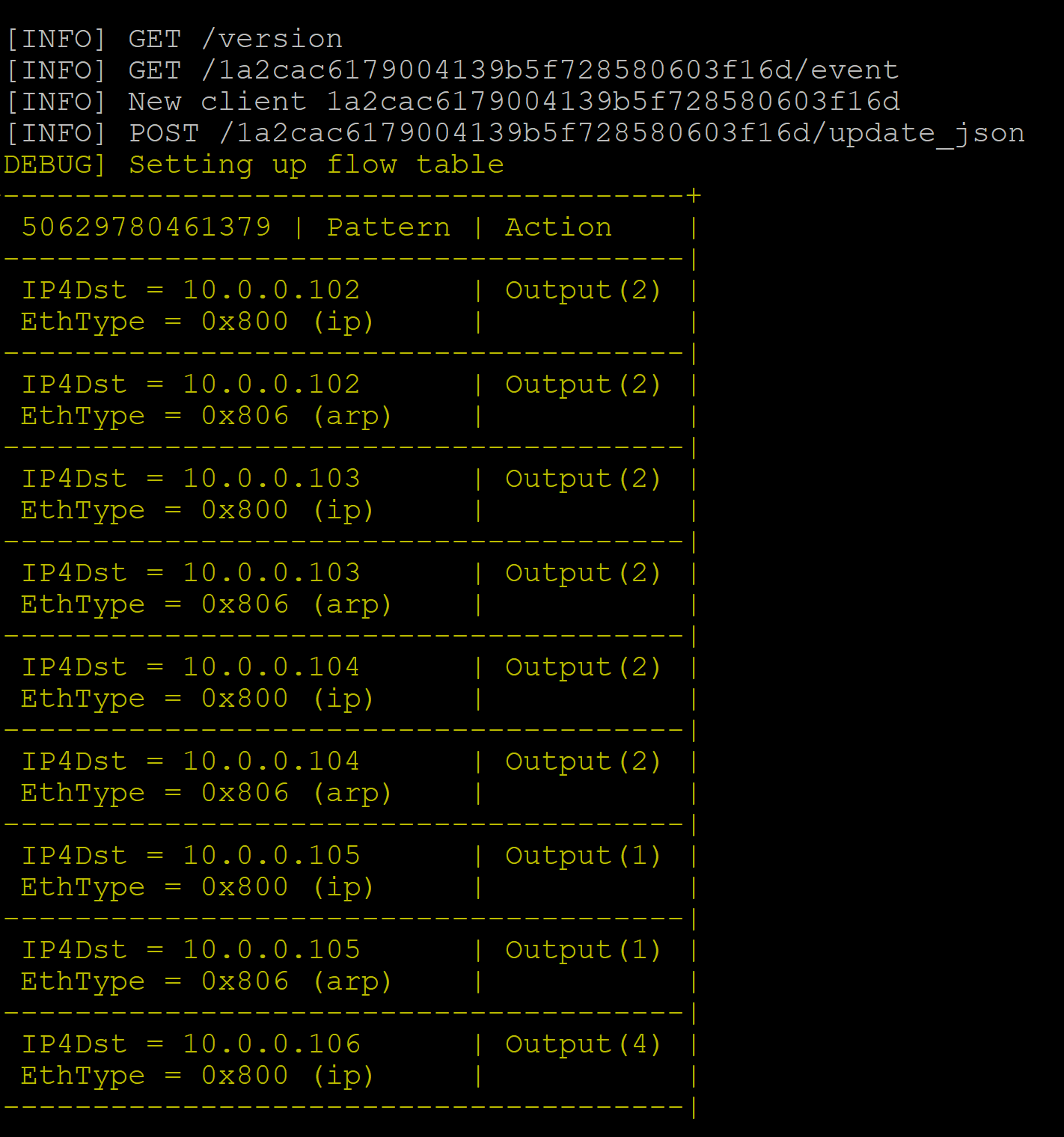
1. In slave switch, for all the servers SetPort should be the port number of the slave switch port number (this need not be same as the root switch, check the eth numbers like you did before for other VM’s) to direct the traffic to root-switch.



1. After saving it, execute the command python openNetwork-ADAPTS.py to run the file and create the bridge. Ensure you’re in the /var/www/public\_html/Dolus\_DDos/app/Python directory when you’re executing the file.

|  |
| --- |
| *cd /var/www/public\_html/Dolus\_DDos/app/Python*  *python openNetwork-ADAPTS.py* |

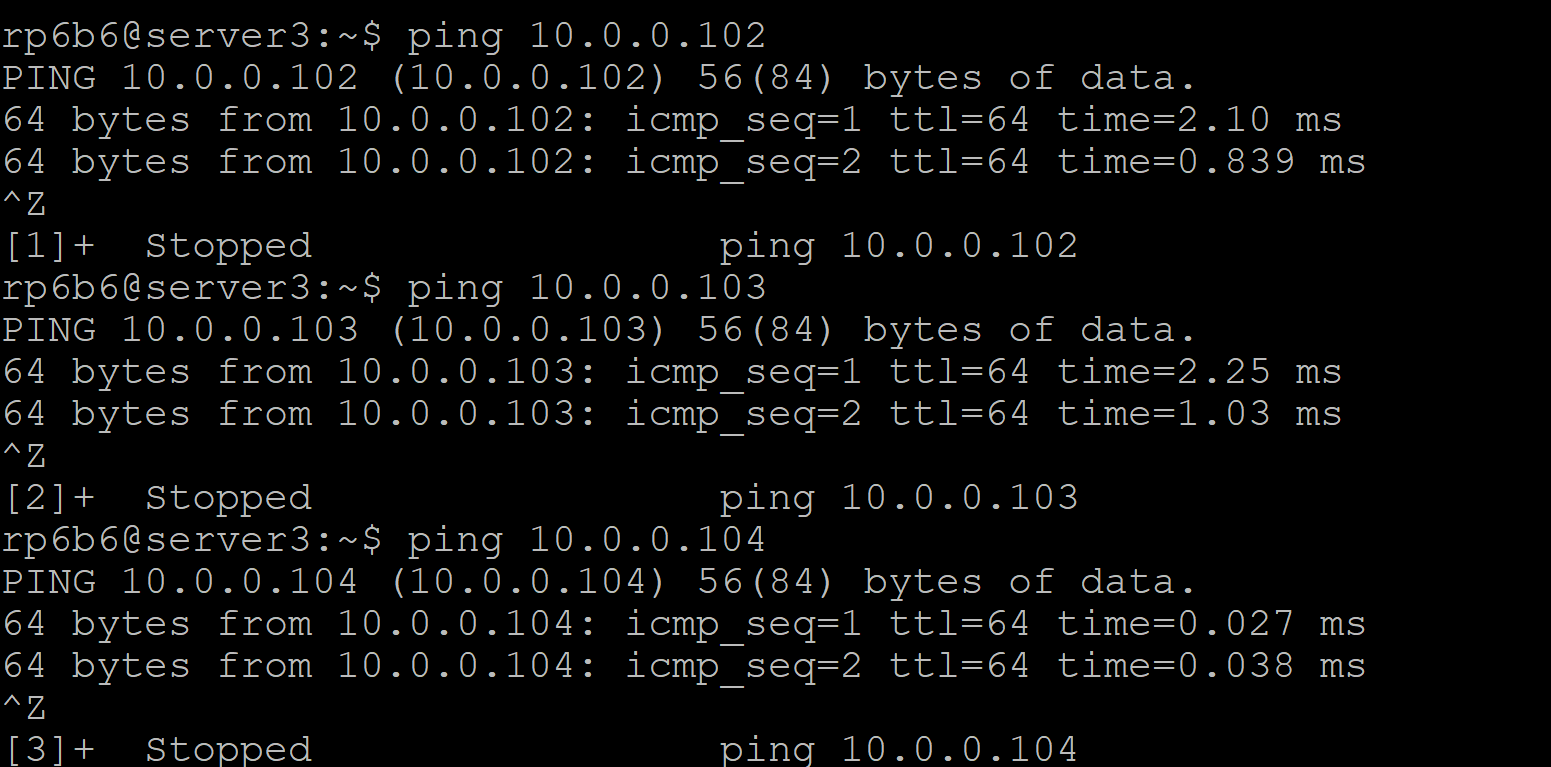
You should see rules table like below.



1. Now that the file has been executed, the bridge has been created to allow the servers connected to the root-switch to be able to communicate with the servers connected to the slave-switch. Log into each device connected to the root-switch and ping the devices connected to the slave-switch. The syntax for ping is ping $ip\_addr, replacing $ip\_addr with the IP address of the device you want to ping. Each ping should succeed, showing that network traffic can be sent between the two switches. Similarly, log into each device connected to the slave-switch and attempt to ping the devices connected to the root-switch. The ping attempts should be successful, and we can now use tshark to capture data and perform the attack using slowhttptest.

**IP address of a VM is the IP address given for Interface to that Virtual Machine.**

1. You should be able to ping one node from another like below



* 1. **Launch a DDoS attack attack**  
     **Note: We assume that TA coordinated GENI team about launching a DDoS attack.**

**Execute both 3.7 and 3.8 simultaneously as we need to capture the bandwidth/network data and packet data while an attack is going on**

**Keep your bandwidth and packet capture scripts running when you launch an attack.**

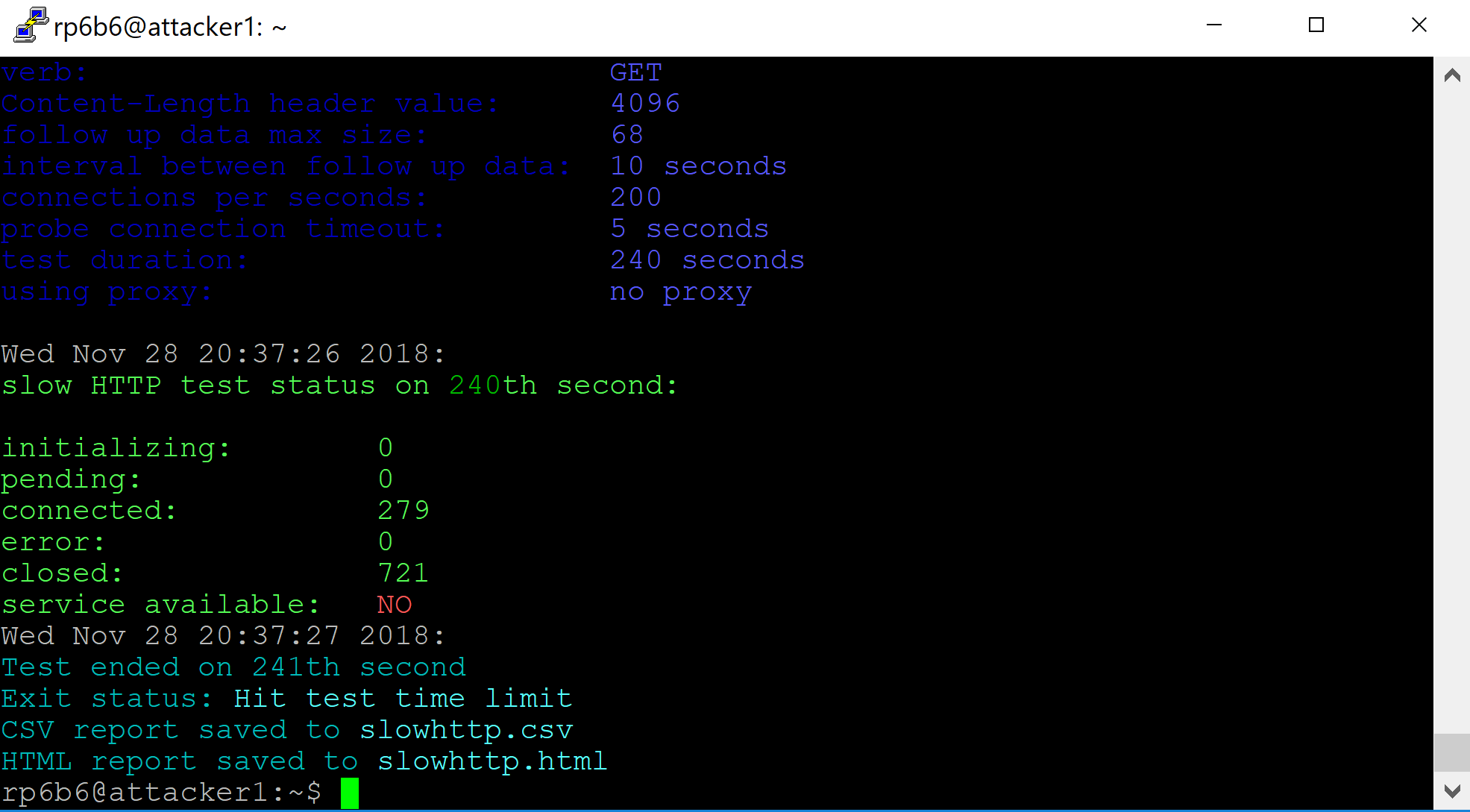
1. Login to Attacker1 Console via SSH using the GENI interface.

After updating the package lists and installing the slowhttptest package, we can now launch the slowhttptest attack. Please execute the command below, replacing $server1IP with the IP address of server1:

|  |
| --- |
| *sudo apt-get -y update*  *sudo apt-get -y install slowhttptest* *slowhttptest -c 1000 -H -g -o slowhttp -r 200 -t GET -u http://$server1IP/* |

***example:*** *slowhttptest -c 1000 -H -g -o slowhttp -r 200 -t GET -u* [10.0.0.102](https://www.dropbox.com/referrer_cleansing_redirect?hmac=eJIxehYKwRrHCgyR3lFGxNlXuEdLclIt6clh3jWjCcA%3D&url=http%3A%2F%2F192.41.233.58%2F)

***You should see the following output after you launch the attack***



* 1. **Capture the packet flow**
     + 1. Install the following commands on both switches:

**Give mysql-server password as ‘root’.** But we will use controllers global user credentials monty and some\_pass later on to login to database.

|  |
| --- |
| *sudo apt-get install -y Tshark*  *sudo apt-get install -y mysql-server*  *sudo apt-get install -y python-mysql.connector*  *sudo apt-get install bwm-ng* |

* + - 1. **Bandwidth Capture**

Open new terminals on both the root switch and slave switch, and execute the following command on both the root switch and slave switch terminals to capture the bandwidth details:

|  |
| --- |
| *rm -f stats.csv*  *bwm-ng -o csv -T sum -F stats.csv* |

* + - 1. **Packet Capture**

And in a different slave switch and root switch terminals run the following Tshark commands to capture the packet flow.

* **SSH in to root switch console**

|  |
| --- |
| *rm -rf root-capture.csv*  *sudo tshark -i eth2 -i eth3 -i eth4 -T fields -e frame.number -e frame.time\_relative -e frame.time\_epoch -e frame.protocols -e frame.len -e eth.src -e eth.dst -e eth.type -e ip.proto -e ip.src -e ip.dst -E header=y -E separator=, -E occurrence=f > ~/root-capture.csv* |

* **SSH in to Slave switch console**

|  |
| --- |
| *rm -rf root-capture.csv*  *sudo tshark -i eth2 -i eth3 -i eth4 -i eth5 -i eth6 -i eth7 -T fields -e frame.number -e frame.time\_relative -e frame.time\_epoch -e frame.protocols -e frame.len -e eth.src -e eth.dst -e eth.type -e ip.proto -e ip.src -e ip.dst -E header=y -E separator=, -E occurrence=f > ~/root-capture.csv* |

**after the attack is launched CTRL+Z and stop the running commands (both packet capture and bandwidth capture script and cancel the data transfer from hosts too).** Now, we can store the data into tables using the following commands

We need to append the switch\_id and trace\_id to the root\_capture.csv. for that we need to execute the following awk command by replacing switchDPID with your switch DPID and trace\_id with the current number of executions. (here trace\_id is the nth no of execution, If it’s the first time it should be 1)

Execute the following commands on both slave switch and root switch consoles opened for **packet capture**.

|  |
| --- |
| *rm -f result.csv*  *awk '{print <switchDPID>,",<trace\_id>,", $0}' root-capture.csv > result.csv* |

***Example****: awk '{print 196040413341508,",2,", $0}' root-capture.csv > result.csv*

To store the packet data captured on a switch into the tables execute the following

command by logging into mysql database on switch

|  |
| --- |
| *mysql -h CONTROLLERIP -u YOURUSERNAME -p --local-infile YOURDBNAME;* |

***Example****:* *mysql -h 72.36.65.106 -u monty -p --local-infile test;*

|  |
| --- |
| *use test;*  *LOAD DATA LOCAL INFILE 'result.csv' INTO TABLE test.packet\_logs COLUMNS TERMINATED BY ',' IGNORE 1 LINES;* |

Now the captured **Bandwidth data** in first terminals of root switch and slave switch including switch\_id is stored in result.csv

|  |
| --- |
| *rm -f outstats.csv*  *awk '{print "<Switch\_id>;", $0}' stats.csv > outstats.csv* |

***example:*** *awk '{print "196040413341508;", $0}' stats.csv > outstats.csv*

To store the packet data captured on a switch into logs tables execute the following

|  |
| --- |
| *mysql -h CONTROLLERIP -u YOURUSERNAME -p --local-infile YOURDBNAME;* |

***Example****:* *mysql -h 72.36.65.106 -u monty -p --local-infile test;*

|  |
| --- |
| *use test;*  *LOAD DATA LOCAL INFILE 'outstats.csv' INTO TABLE test.logs COLUMNS TERMINATED BY ';' (switch\_id, unixtimestamp, port\_id, tx\_bytes, rx\_bytes, total\_bytes, tx\_packets, rx\_packets,total\_packets,tx\_errors,rx\_errors);* |

* 1. **Detecting and Deceiving the attacker using QM**

**Detect an Attack**

1. Now, log on to the controller via SSH and execute the following commands to detect the suspiciousness scores on each switch:

|  |
| --- |
| *cd /var/www/public\_html/Dolus\_DDos*  *sudo apt-get install python-mysqldb*  *sudo pip install MySQL-python python app/Python/main.py* |

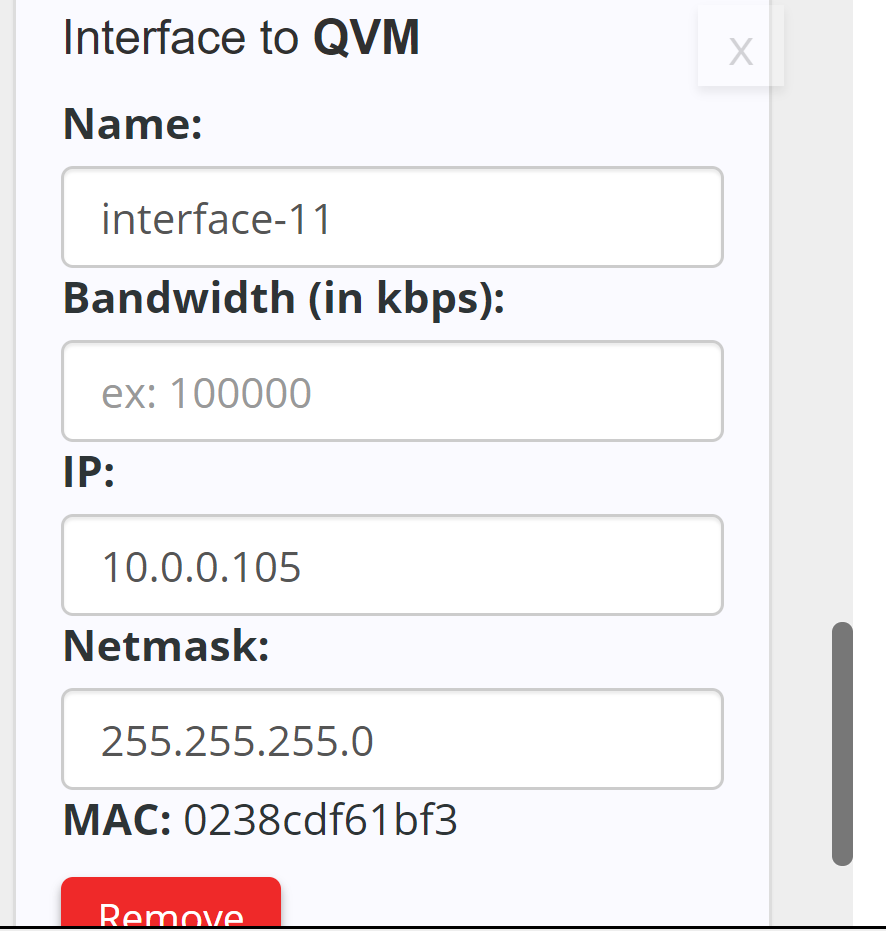
**If you get any error on no module named MySQLdb, try installing Mysqldb for python.** **If you face any issues while executing main.py, you might have to delete the data from your suspiciousness\_scores table and reexecute main.py(as trace\_id’s will be different if you execute without deleting the data from suspiciousness\_scores and you might get unexpected results)**   
The main.py Python script calls Python scripts to calculate the suspiciousness scores for each device.

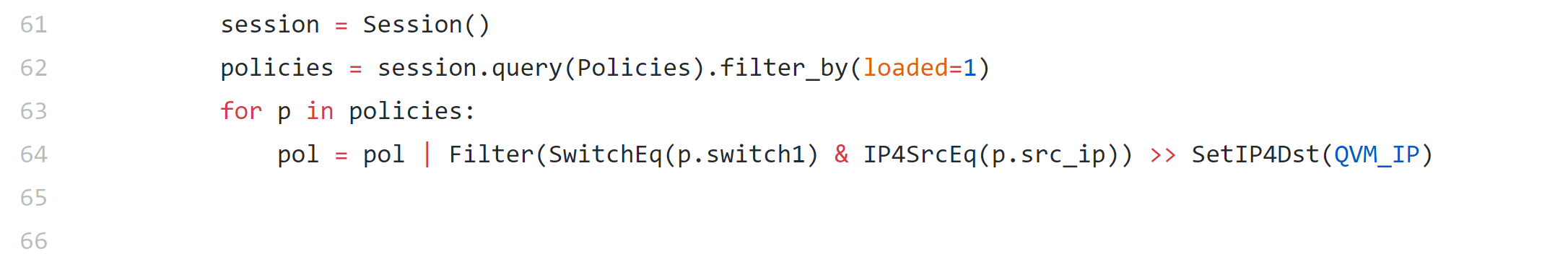
**Redirect the attacker traffic**

1. At this point, we’re going to deceive the attacker. On the controller, execute the following commands to change the directory to the one containing the Python scripts and edit the openNetwork-ADAPTS.py file:

|  |
| --- |
| *cd /var/www/public\_html/Dolus\_DDos/app/Python sudo nano openNetwork-ADAPTS.py* |

Navigate to line 64 and change the IP address inside of SetIP4Dst(), ensuring that the IP address you change it to is the IP address of the QVM server. You can find the IP address like below. **IP address of a VM is the IP address given for Interface to that Virtual Machine.**





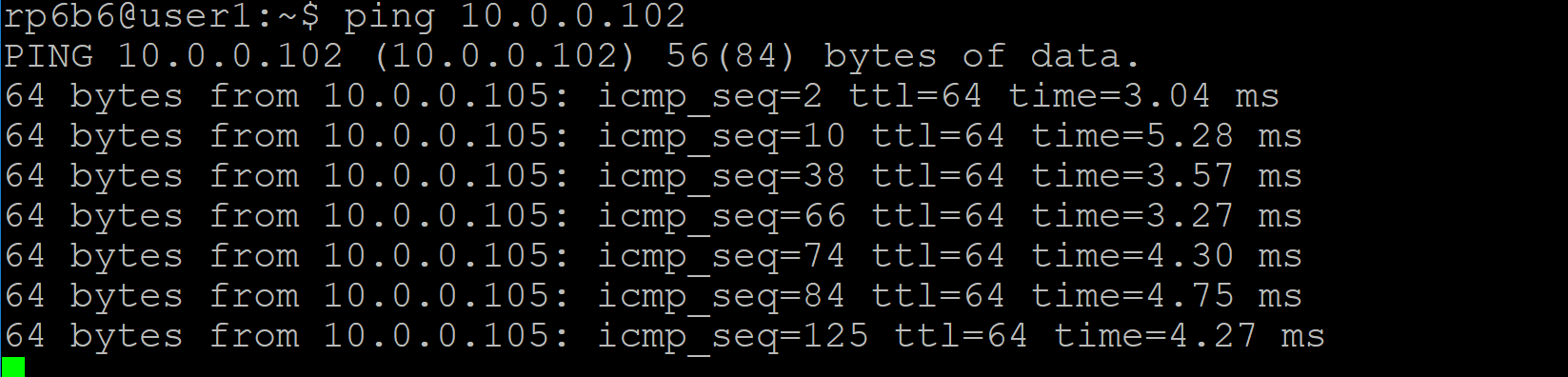
Then, save and quit (in nano, the syntax to save is Ctrl+O, then to quit is Ctrl+X). Changing the line above allows the script to reset the network with a custom policy that redirects Attacker Traffic to the QVM. Attacker won’t be able to ping any other device as we set attacker destination to QVM.

1. Run below python script to redirect the attacker to QVM.

|  |
| --- |
| *python openNetwork-ADAPTS.py* |

Whenever an attack is detected, A policy will be created to redirect the attacker to qvm. You can see the new Policy created in Policy table for the srcIP. Take a screenshot of the new Policy record created in policy table.

You should see the following flow on your attacker console.

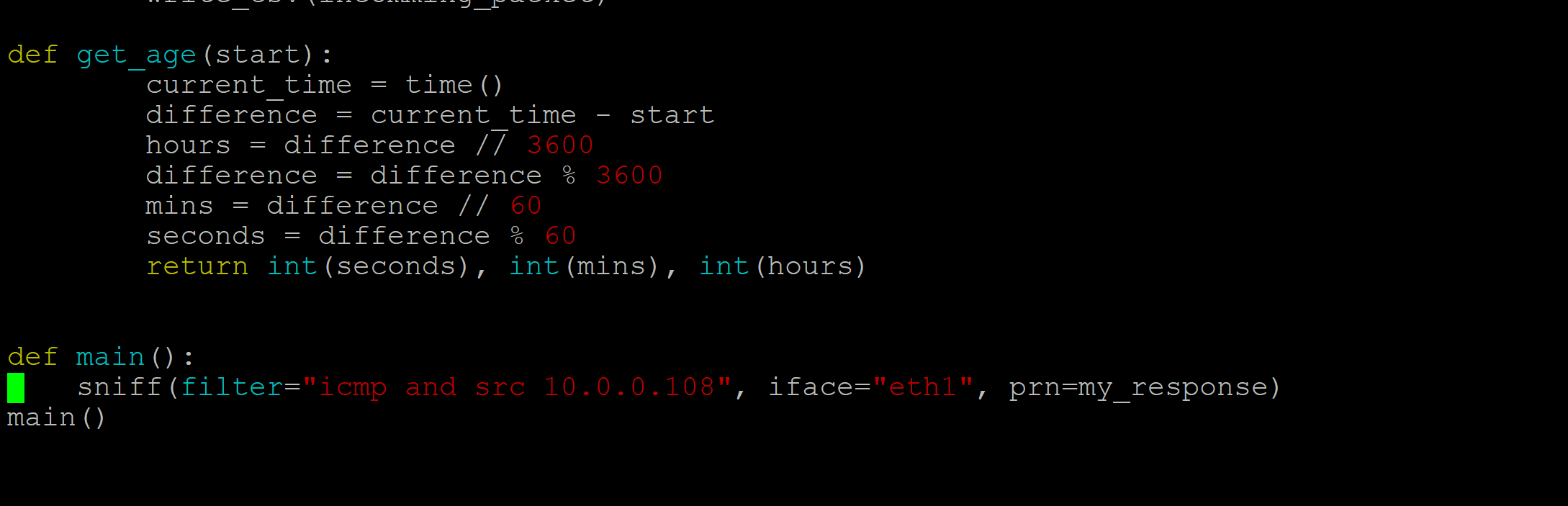


**Deceive the attacker**

1. Run the following pretense.py on your QVM Console(You can find this file in Dolus-Lab 2 modules), this python script uses Scapy to send fake responses to attacker to deceive him

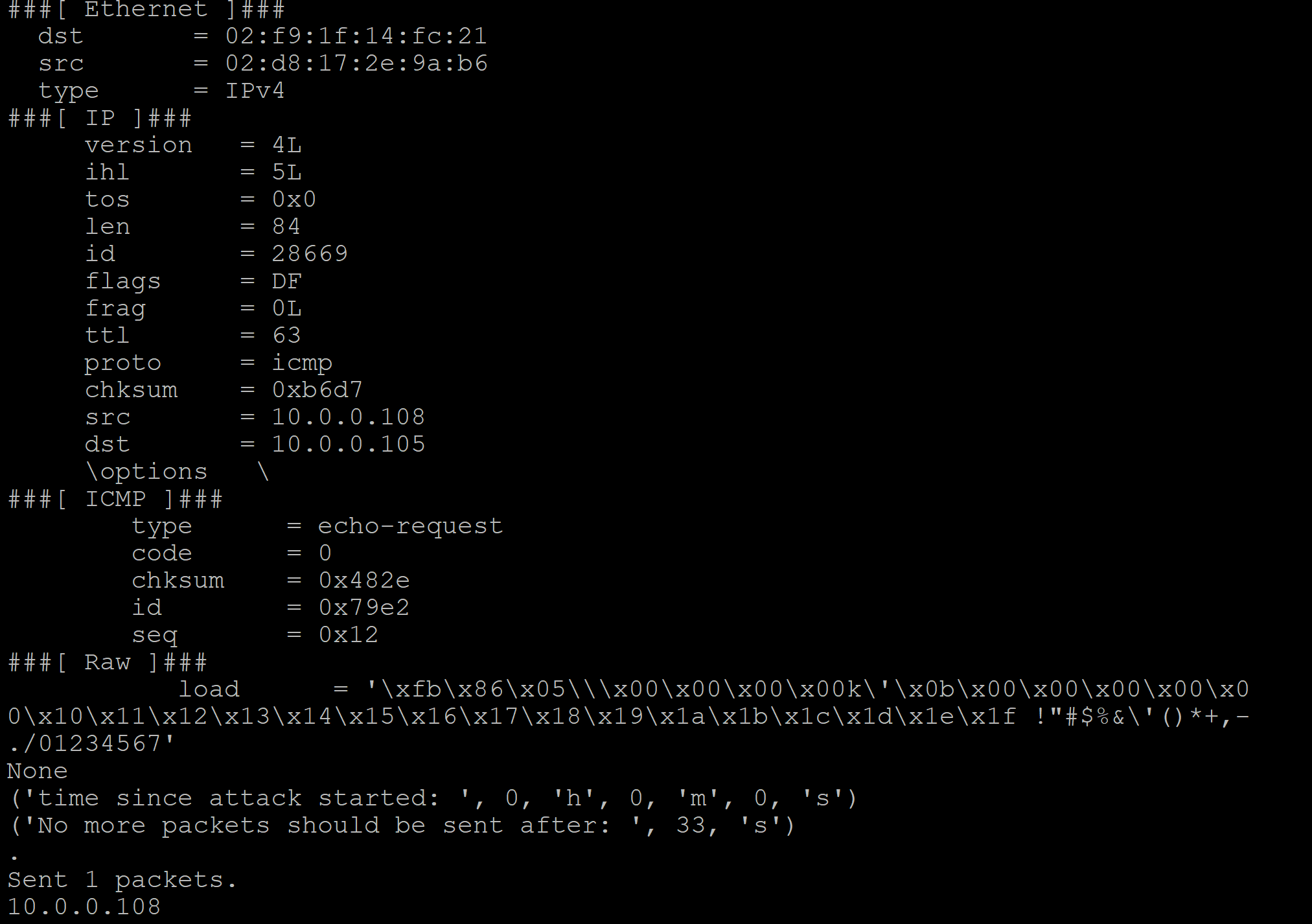
|  |
| --- |
| *sudo apt-get update*  *sudo apt-get install scapy* |

Open you pretense.py and edit your attackerIP as shown below And then execute pretense.py



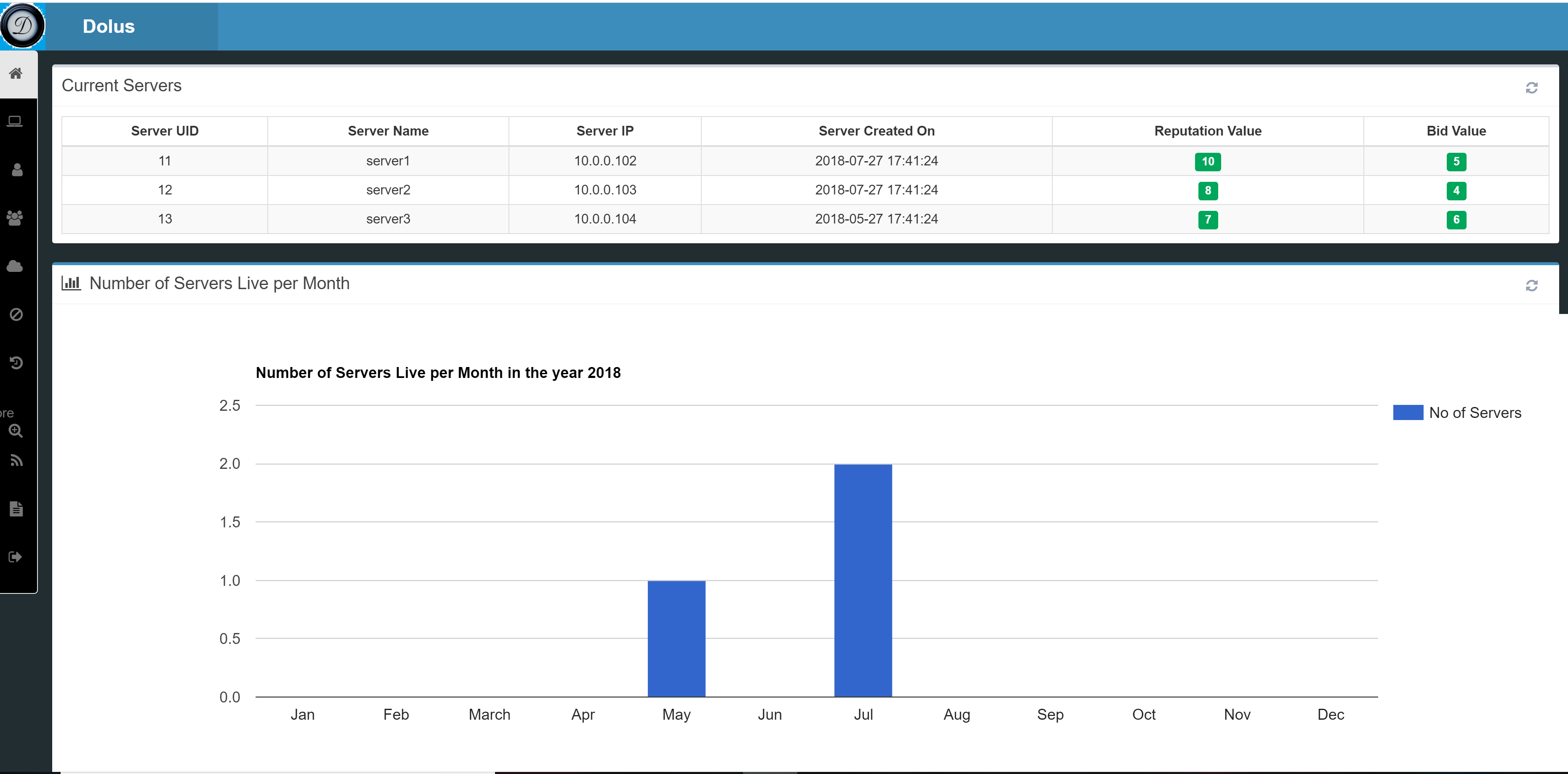
|  |
| --- |
| sudo python pretense.py |

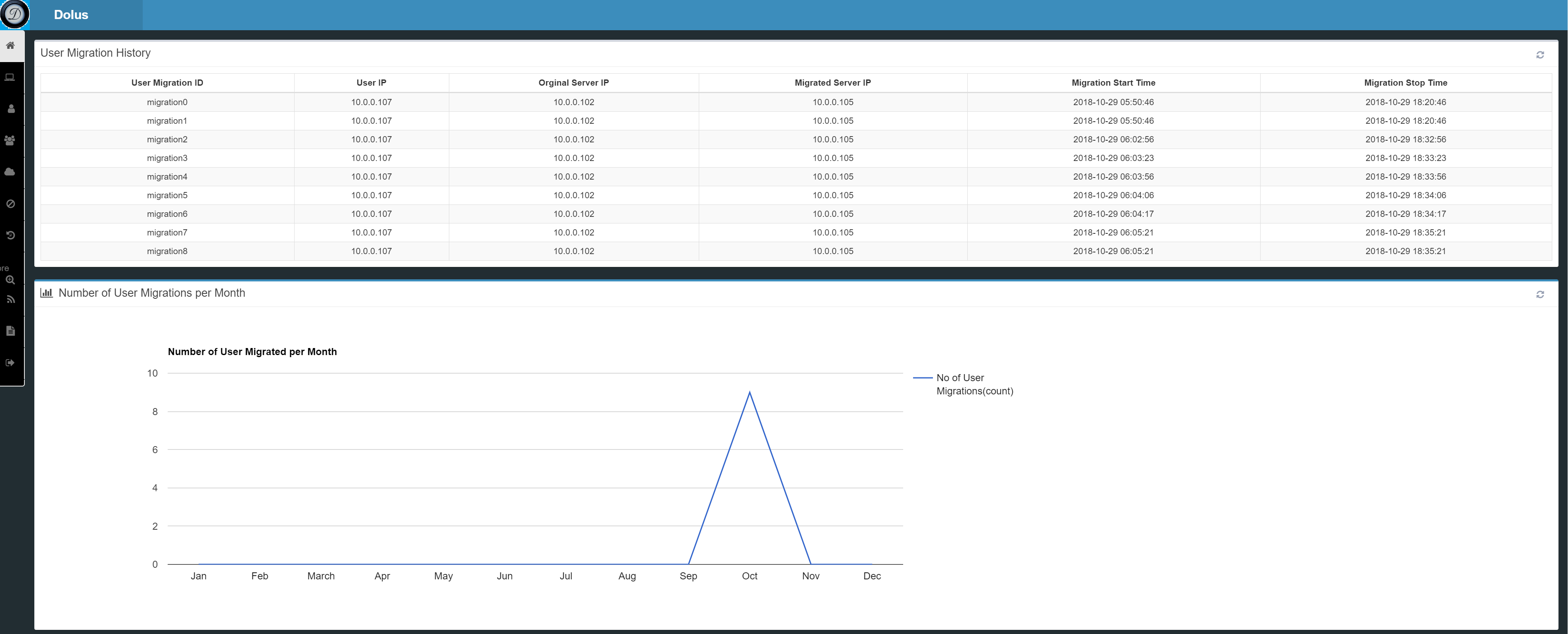
When you ping server from attacker, you will see the following output in QVM console

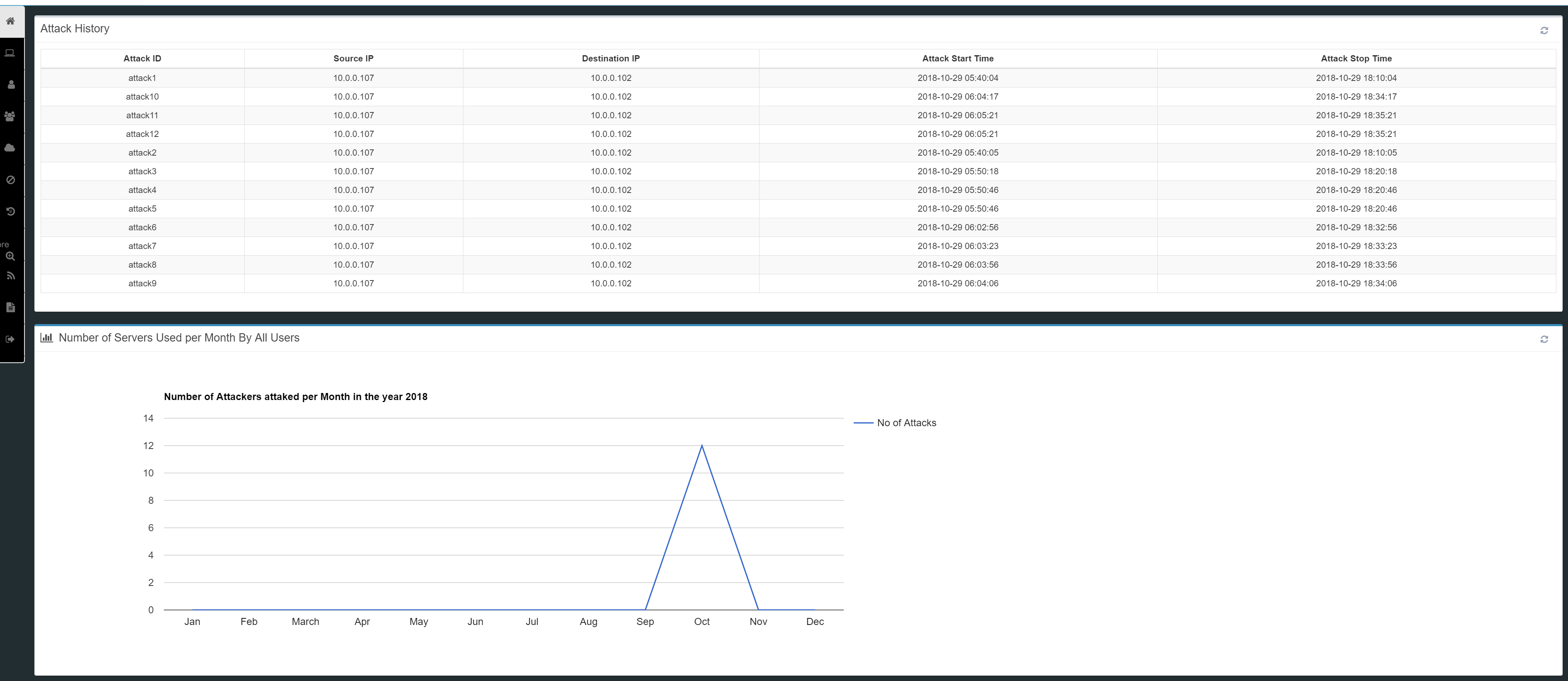


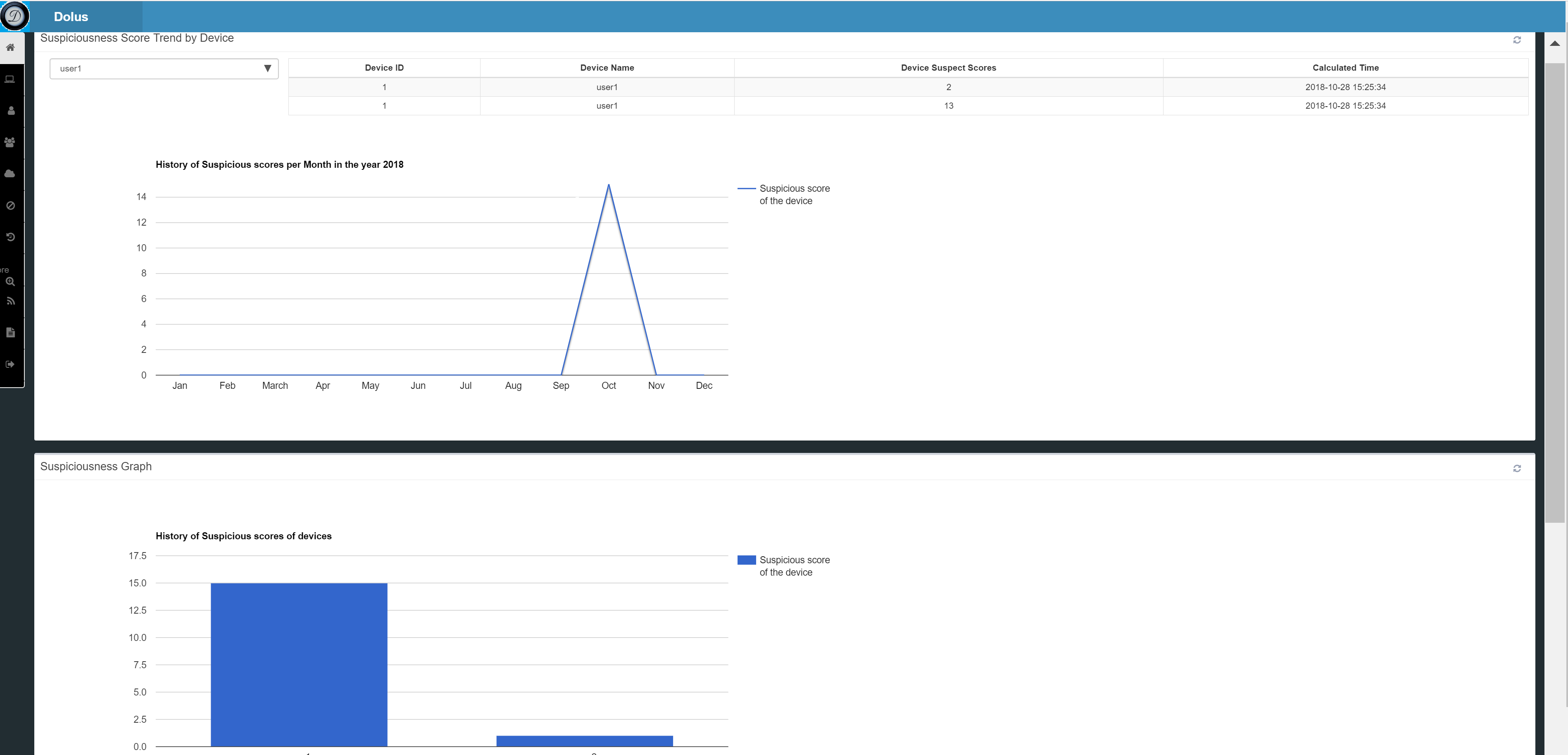
* 1. **Observing the attacker, user and server information in the website.**

**(Don’t panic if your graphs little different, it might change according to the data incoming or connections made)**









1. **What needs to be turned in for Grading?**
   * 1. What is Distributed Denial of service attacks? Suppose a web application is hosted on Server1 and several users are trying to access the Web application. Explain briefly the following scenarios?
2. What will happen when many users try to refresh the web browser in which they are accessing the webpage over and over on different computers at once. Explain me briefly what type of DDoS attack is possible in this scenario and a possible solution in your own words?
3. A User broadcasted a huge volume of TCP, UDP and ICPM  packets with the intended victim's [spoofed](https://en.wikipedia.org/wiki/IP_address_spoofing) source IP to a [computer network](https://en.wikipedia.org/wiki/Computer_network) using an IP [broadcast address](https://en.wikipedia.org/wiki/Broadcast_address). What can happen when most devices on this network respond to this by sending a reply to the source IP address. Give a possible solution in your own words?
   * 1. What is an Memcached DDoS attack? Explain in detail, how does the code repository GitHub was [taken off air](https://www.theregister.co.uk/2018/03/01/github_ddos_biggest_ever/) in a 1.3Tbps denial of service attack ? What are the implications to the organization running memcache servers and what can be the amplification of the attack? What is the cost and expertise needed to setup a cloud configuration that can mitigate the memcache attack?
     2. Describe the methodology used in detecting an attacker in step 3.8 and 3.9
     3. Take screenshots of the graphs of Network flow, bandwidth flow and switches from AdminUI and describe the network abnormality in your own words.
     4. Explain in detail, what surprised you about the Dyn.Inc Attack? What caused the attack Human error, technology convergence or Infrastructure? What’s new that we can learn to defend DDoS attacks at this scale? Write Scapy commands to recreate the attack in GENI testbed and Frenetic commands to apply pretense. What’s preventing the attacker from going to the new URL which legitimate users are going?
     5. What are NETKAT Policies and how can we implement policies for the security of the network? Suppose all the traffic to ServerS1 through root\_switch is directed to port 2 as given by the following command:

*pol = Filter (SwitchEq(root\_switch) & IP4DstEq ("10.10.2.2")) >> SetPort(3)*

Similarly, write commands to redirect the attacker traffic to a Quarantine machine and block the traffic from attacker to server1 using NETKAT Policies where IP address of QVM is 10.10.5.2 and the attacker IP address is 10.10.10.1.

* + 1. How would you modify NetKat policies to include an additional slave switch connected to the root switch? How would the policy change to allow data to transfer between a device on the first slave switch and a device on the second slave switch?
    2. **Submit your Controller IP along with username and password**