



# Final Report

CS 4000

Gregory Harrison  
6-30-2015

RSA Encryption .....	2
Suggested Attacks using Mininet.....	4
Implemented attacks .....	4
List Attacks .....	5
Figures.....	7

# RSA

The RSA encryption is the modern standard for sending secure information. RSA is an algorithm based on the modular arithmetic of prime numbers. This encryption relies not on just a single key but on two different keys. To show someone the information that is encrypted you'll just have to tell them the public key, never tell anyone your private key (think about your private key like it's a password that you'll have to keep safe). In a general term, the RSA encryption you choose 2 prime numbers,  $p$  and  $q$ , and multiply them to produce a number  $N$ . The number  $N$  will be the modulus of the algorithm and will be made public. In a complex encryption this will be okay because  $p$  and  $q$  are both prime number and very large numbers. With  $p$  and  $q$  we can now make our public and private keys with the given information.

## RSA Algorithm

1. Generate two random primes,  $p$  and  $q$
2. Compute  $N = p * q$
3. Compute  $Z = (p-1)*(q-1)$
4. Choose an integer  $e$ ,  $1 < e < Z$
5. Choose an integer  $d$ ,  $(d*e) \% Z = 1$

The public key is  $(e, N)$  and the private key is  $(d, N)$ .

$N$  is known as the modulus.

$e$  is known as the public exponent or encryption exponent or just the exponent.

$d$  is known as the secret exponent or decryption exponent.

Encryption – With the given information you are able to compute your encryption.

*Let's say that the message,  $M$ , is what you want to encrypt.*

*Using our public key  $(e, N)$  we compute  $C = M^e \bmod N$*

With our message (plaintext),  $M$

Compute the ciphertext with  $C = M^e \bmod N$

Now we can send our secure ciphertext out

Decryption – With given the ciphertext and the private key we can find our message

With the private key,  $(d, N)$  we find our message

With the formula  $M = C^d \bmod N$

## Real World application

1. Alice knows that Bob wants to send her a message, so she selects two prime numbers. Let's say she picks  $p = 3$  and  $q = 11$ .

- a. Alice then multiplies  $p$  and  $q$  together to get the number  $N$ .
  - b.  $P * q = 3 * 11 = 33$
  - c. So Alice now has  $N = 33$
2. Alice need to find her  $Z$ 
  - a. She then computes  $Z = (p - 1) * (q - 1)$
  - b.  $Z = 2 * 10 = 20$
3. Alice can now can compute her  $e$  such that it is greater than 1 and less than 8.
  - a. Alice chooses  $e = 7$
4. Alice then gives the numbers  $N = 33$  and  $e = 7$  to Bob.
5. With these two numbers  $N$  and  $e$ , Bob can now encrypt his message
  - a. Bob encrypts his message in this can is an age of 2.
  - b. He takes this age and puts it to the power of  $e$ , which he knows is 7
    - i.  $2^7 = 128$
  - c. Bob takes this number and divides it by  $N$  and is only interested in the remainder portion of the division so he mod the equation
    - i.  $C = 128 \% 33 = 29$
  - d. So Bob encrypted 22 as  $C = 29$ , which is the number that he sends to Alice.
6. Alice received the number  $C = 29$  from Bob and she needs to decrypt it to get his age.
7. To decrypt his age Alice goes back the computing  $Z$ 
  - a. Alice needs to find a number  $d$  such that  $(d * e) \bmod Z = 1$
  - b. Alice chooses her  $d = 3$
  - c. Alice computes the message by take the encrypted message  $C$  and putting it to the power of  $d$ .
    - i.  $C^d$
    - ii.  $29^3 = 24,389$
  - d. Alice then divides that number by  $N$  but is only instead in the reminder so she uses mod
    - i.  $C^d \bmod N = \text{Message}$
    - ii.  $24,389 \bmod 33 = 2$

Alice Knows	Bob Knows	Everyone else can know
-------------	-----------	------------------------

P, q, N, Z, e, d, C	N, e, C	N, e, C
---------------------	---------	---------

## Suggested Attacks using Mininet

### DHCP masquerade Attack

In in mininet, you create a switch with 3 host and 1 switch. H1 one will be the victim that is just trying to connect to the Wi-Fi and access a certain website which will make a DHCP (Dynamic Host Configuration Protocol) request which is a network protocol that enables server to automatically assign an IP address to the given computer. The next host is a DHCP is a good Server which provides the correct information. Then you have your hacker which is connected directly to the switch. When h1 makes a request it forwarded to both the DHCP and the hacker , and the hacker responds first and it DHCP offers is accepted by the victim.

### Eavesdropping Attack

In Mininet, if there are two people (A and B) send packets back and forth. In another terminal if we run a tcpdump to eavesdrop the traffic between person A and B. Since the switch between these three people have already learned about the addresses of person A and B, person C will not receive any packet from those two. We simulate person C running some Ethernet packets with randomly generated source MAC address to overflow the switch. Switch will start showing traffic of person A and B and they should start showing when person C does another tcpdump.

## Implemented attacks

### DOS attack

I tried making different DOS attack neither of them would work for me I've tried replicating the DHCP attack code that I found and kept receiving errors in that code shown (Figure 5). I am lost with this one I don't know how to start on my own. I've tried reading the code shown below but that gives me the same no results it just ran in the background (Figure 6).

### Ping Flood

For the ping flooding, I created a sudo mn -topo=single, 4 topology, which is 4 host and one switch. Then I controlled the ping in which each host sent an endless about of packets of size 65500 bytes to the fourth host. First I record the fourth host send 10 packets out before I flooded the host with request (Figure 1). Then after sending an endless amount of packets to the fourth host I resent 10 packets from host 4 and the time of each packet send was increased by a couple of seconds(Figure 2). I also recorded the different notes that showed up while flooding the host there were retransmission,

duplicate acknowledgements received, and an undecoded message (Figure 3). Lastly I recorded the summary during the time of how many packets were sent and the number of bytes received (Figure 4).

## List Attacks

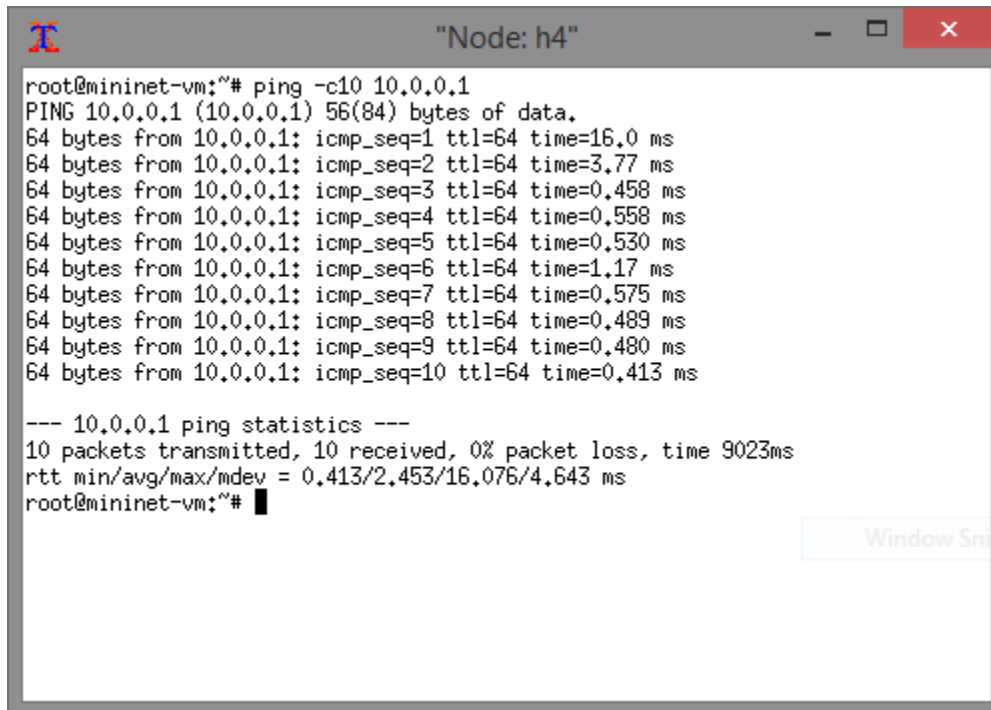
- Basic attacks
  - Physical theft - someone steals network hardware such as wires, hubs, or other equipment that keeps the network running
  - Subversion - someone modifies or otherwise takes over part of the network so that it enables an attack. **Threats involves physical or logical changes to network components (What do they mean by components )**
    - Example an attacker might reroute traffic to allow its interception.
  - Disclosure - an attacker's computer intercepts copies of network data intended for others. While this may pose no risk for a lot of network traffic. ( This type of eavesdropping may yield passwords or other data that enables a more serious attack)
  - Forgery - someone constructs a bogus message or modifies a legitimate message as part of an attack.
    - Bogus order could send merchandise without collecting payment
  - Masquerade - a person tricks the network into sending messages claiming to be originated by someone else. In the networking environment, this behaves like a particular type of forgery
  - Denial of service - an attack that makes some or all of the network unusable. Typical attacks either flood parts of the network with traffic or render network components unusable
- Worms like viruses copy themselves to other places and spread from those other place. While viruses may infect drives and applications programs worms infect host computers.
- Inside attacks
  - Disclosure – theft of trade secrets
  - Masquerade – fraud, social engineering
  - Service loss – extortion, vandalism
  - Subversion – fraud, rootkits
  - Physical theft – equipment theft, laptop theft
- Distributed denial of service
- Communication attacks
  - Passive attack – eavesdropping without interfering with communication
  - Active attack – a network attack in which he/she forges network traffic.
- Attacks on Protocol
  - Ping Flood
    - In a ping flood one or more host conspire to flood a victim with ping request
    - The flood of request keep the victim protocol stack and network very busy
  - Smurf attack

- This is a variant of the ping flood in which a single attacker tricks other host into attacking the victim
    - The results in a ping flood engulfing the victim in a DDOS attack
  - Ping of death
    - The ping of death is not actually a protocol weakness it causes by mishandling of too large messages so it was a form of buffer overflow error
    - Redirection attack
- Attacks on RSA
  - Small private key attacks – if the private key  $d$  is small there is a shortcut to recovering it
  - Timing attack – private key's value directly affects execution time of the RSA algorithm
  - Chosen ciphertext
  - Bogus prime numbers – RSA uses extremely large prime numbers to construct a key
- Birthday Attacks
- Publishing public keys
  - Man in the middle attack – an attacker actively intercepts all traffic
  - Bucket brigade attack
- TCP/IP attacks
  - SYN flood attack
    - The attackers sent a series of SYN packets to the victim each specifying a different socket. The source address were usually forged because there was no need to process the response. Each SYN packet produce half open connection. The connection persisted until the protocol stack timed out
  - Source Routing Attack
    - This is a clever variant of the redirection attacked just noted: the IP header contains an option for source routing which allows the sender to direct the traffic through a series of host. The attacker forges a packet from a trustworthy host and puts the attacker's host on the packet's route. The attacker directs the packet at the victim. The victim typically will respond using the source route provided in the original packet. This takes the packet to the attacker's host enroute to the trustworthy host. The attacker's host simply processes the packet and doesn't forward it further
  - IP Spoofing Attacks
    - IP spoofing refers to any attack that forges the sender's IP address
- Attacking DNS
  - Cache poisoning: a resolver receives a bogus response to a DNS query. All subsequent queries will receive the wrong information and redirect connections to the wrong IP addresses
  - DOS attack on major DNS servers: Attackers try to disable DNS service in parts of the internet by attacking major DNS servers
  - DOS attack using a shared Resolver: an attacker transmits numerous bogus DNS queries to the shared resolver
- Email based attacks
  - Connection based attack – incident in which someone sniffs or intercepts another's email session
  - Spam – unsolicited email
  - Phishing – email that tries to trick the reader into disclosing confidential id

- Email viruses – email that contains malicious software that spreads via email

## Figures

Figure 1



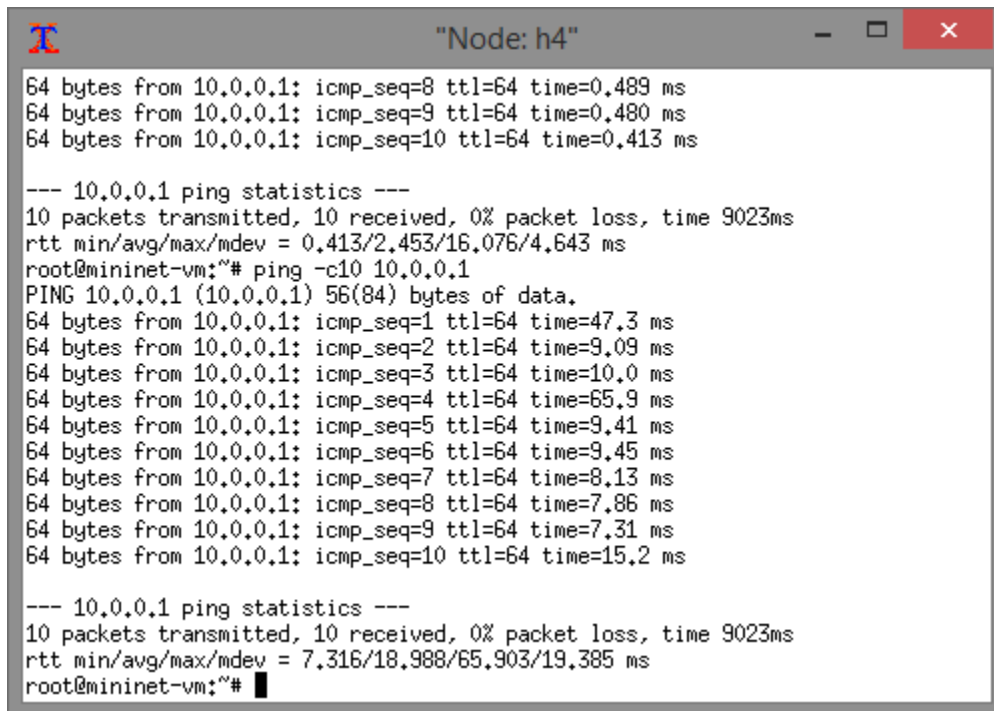
A terminal window titled "Node: h4" showing a ping command and its output. The output displays 10 successful ping results to 10.0.0.1, followed by a summary of the statistics.

```
root@mininet-vm:~# ping -c10 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=16.0 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=3.77 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.458 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.558 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.530 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=1.17 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.575 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.489 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.480 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.413 ms

--- 10.0.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9023ms
rtt min/avg/max/mdev = 0.413/2.453/16.076/4.643 ms
root@mininet-vm:~#
```



Figure 2



```
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.489 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.480 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.413 ms

--- 10.0.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9023ms
rtt min/avg/max/mdev = 0.413/2.453/16.076/4.643 ms
root@mininet-vm:~# ping -c10 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=47.3 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=9.09 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=10.0 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=65.9 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=9.41 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=9.45 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=8.13 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=7.86 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=7.31 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=15.2 ms

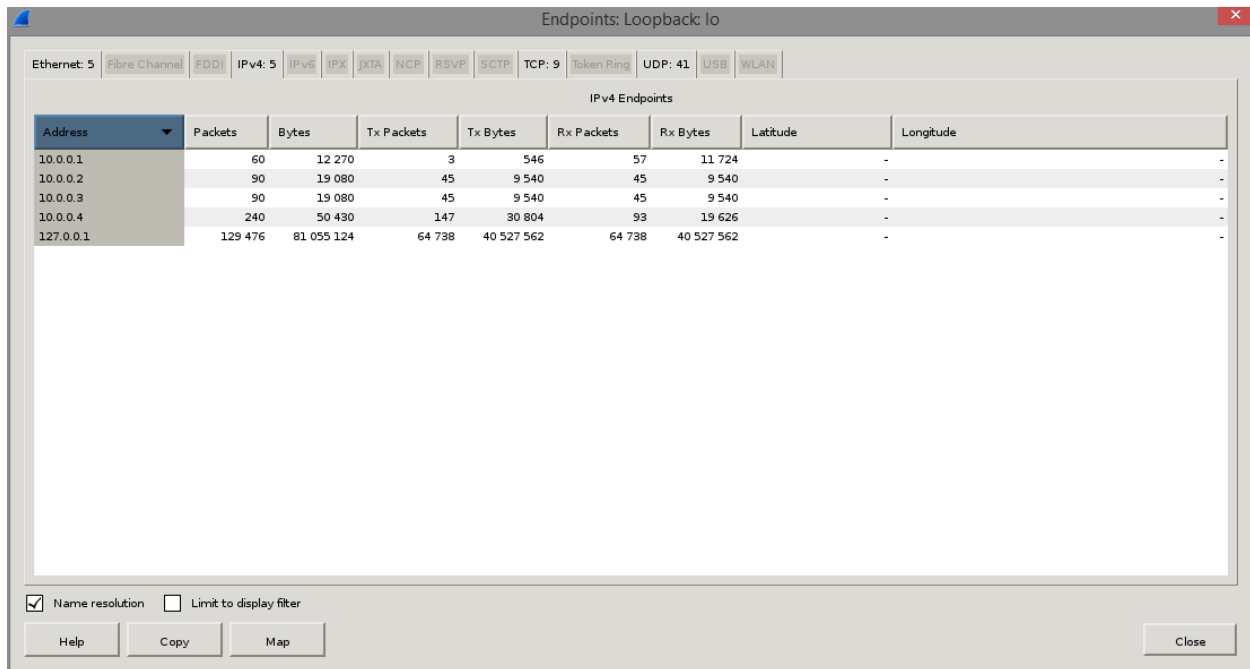
--- 10.0.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9023ms
rtt min/avg/max/mdev = 7.316/18.988/65.903/19.385 ms
root@mininet-vm:~#
```

Figure 3



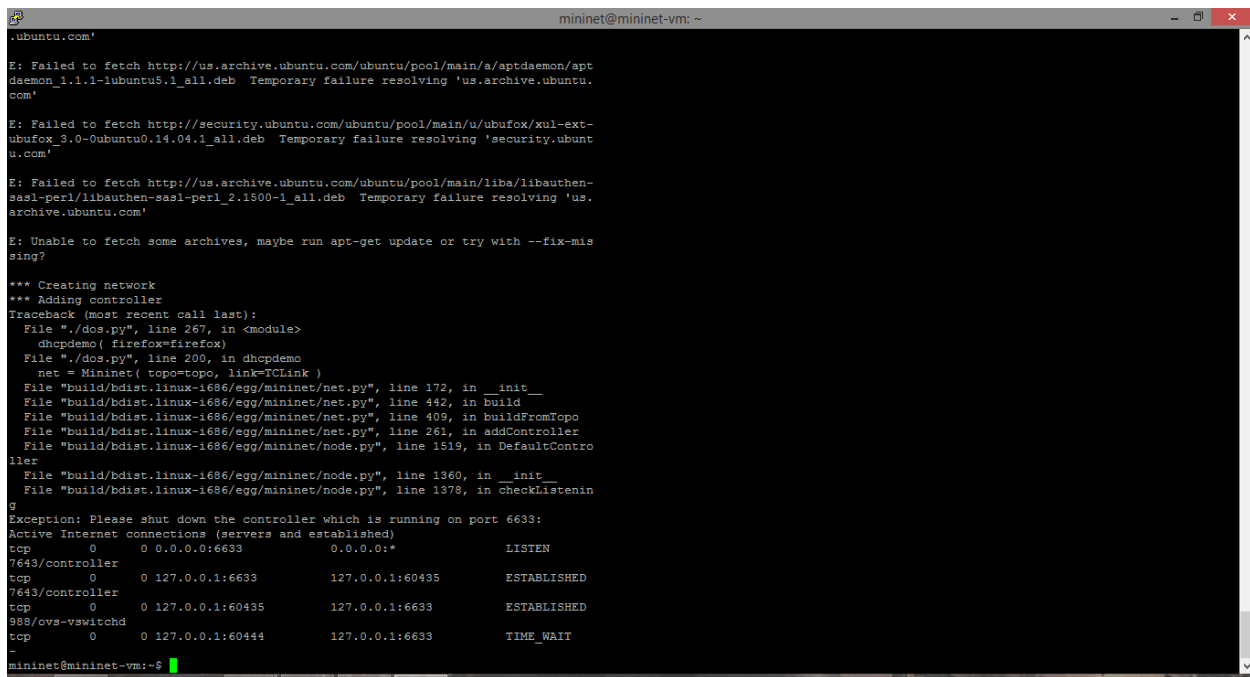
Group	Protocol	Summary	Count
Sequence	TCP	Retransmission (suspected)	57
Sequence	TCP	Duplicate ACK (#1)	7
Undecoded	DCERPC	No bind info for interface Context ID:27393	1
Packet:		25503	1
Response	DCERPC	Fault: Unknown (0x001901cf)	1
Packet:		29580	1

Figure 4



Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Latitude	Longitude
10.0.0.1	60	12 270	3	546	57	11 724	-	-
10.0.0.2	90	19 080	45	9 540	45	9 540	-	-
10.0.0.3	90	19 080	45	9 540	45	9 540	-	-
10.0.0.4	240	50 430	147	30 804	93	19 626	-	-
127.0.0.1	129 476	81 055 124	64 738	40 527 562	64 738	40 527 562	-	-

Figure 5



```
mininet@mininet-vm: ~  
ubuntu.com'  
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/s/aptdaemon/apt  
daemon_1.1.1-ubuntu5.1_all.deb Temporary failure resolving 'us.archive.ubuntu.  
com'  
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/u/ubufox/xul-ext-  
ubufox_3.0-0ubuntu0.14.04.1_all.deb Temporary failure resolving 'security.ubunt  
u.com'  
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/liba/libauthen-  
sasl-perl/libauthen-sasl-perl_2.1500-1_all.deb Temporary failure resolving 'us.  
archive.ubuntu.com'  
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis  
sing?  
*** Creating network  
*** Adding controller  
Traceback (most recent call last):  
  File "./dos.py", line 267, in <module>  
    dhcpdemo( firefox=firefox)  
  File "./dos.py", line 200, in dhcpdemo  
    net = Mininet( topotopo, link=ICLink )  
  File "build/bdist.linux-1686/egg/mininet/net.py", line 172, in __init__  
  File "build/bdist.linux-1686/egg/mininet/net.py", line 442, in Build  
  File "build/bdist.linux-1686/egg/mininet/net.py", line 409, in buildFromTopo  
  File "build/bdist.linux-1686/egg/mininet/net.py", line 261, in addController  
  File "build/bdist.linux-1686/egg/mininet/node.py", line 1519, in DefaultContro  
ller  
  File "build/bdist.linux-1686/egg/mininet/node.py", line 1360, in __init__  
  File "build/bdist.linux-1686/egg/mininet/node.py", line 1378, in CheckListenin  
g  
Exception: Please shut down the controller which is running on port 6633:  
Active Internet connections (servers and established)  
tcp        0      0 0.0.0.0:6633          0.0.0.0:*            LISTEN  
7643/controller 0      0 127.0.0.1:6633       127.0.0.1:60435      ESTABLISHED  
7643/controller 0      0 127.0.0.1:60435      127.0.0.1:6633       ESTABLISHED  
988/ovs-vsitchd 0      0 127.0.0.1:60444      127.0.0.1:6633       TIME_WAIT  
mininet@mininet-vm:~$
```

Figure 6

```
Python 2.7.8: dos_main.py - \CS6250-master\CS6250-master\assignment-8\dos_main.py
File Edit Format Run Options Windows Help

# import pyresonance stuff
from ..drivers.sflow_event import *
from ..globals import *

# import other files from this assignment
from dos_fsm import DDoSFSM
from dos_policy import DDoSPolicy

HOST = 'localhost'
PORT = 8008

def main(queue):

    # Create FSM object
    fsm = DDoSFSM()

    # Create policy using state machine
    policy = DDoSPolicy(fsm)

    # Create an event source (i.e., SFlow)
    sflow_event = SFlowEvent_T(fsm.default_handler, HOST, PORT)

    sflow_event.set_max_events(10)
    sflow_event.set_timeout(60)

    groups = {'external': ['0.0.0.0/0']}
    flows = {'keys': 'ipsource,ipdestination', 'value': 'frames'}
    threshold = {'metric': 'ddos', 'value': 10}
    message = {'event_type': 'ddos', 'message_type': 'state', 'message_value': 'ddos-attacker', \
        'flow': {'dstip': None, 'protocol': None, 'srcmac': None, 'cos': None, 'vlan_pcp': None, 'dstmac': None, \
        'import': None, 'ethtype': None, 'srcip': '10.0.0.1', 'dstport': None, 'srcport': None, 'vlan_id': None}}

    sflow_event.set_groups(groups)
    sflow_event.set_flows(flows)
    sflow_event.set_threshold(threshold)
    sflow_event.set_action(message)

    sflow_event.start(queue)

    return fsm, policy
```

Ln: 36 Col: 0