Tamania Choudhury

Penetration Testing

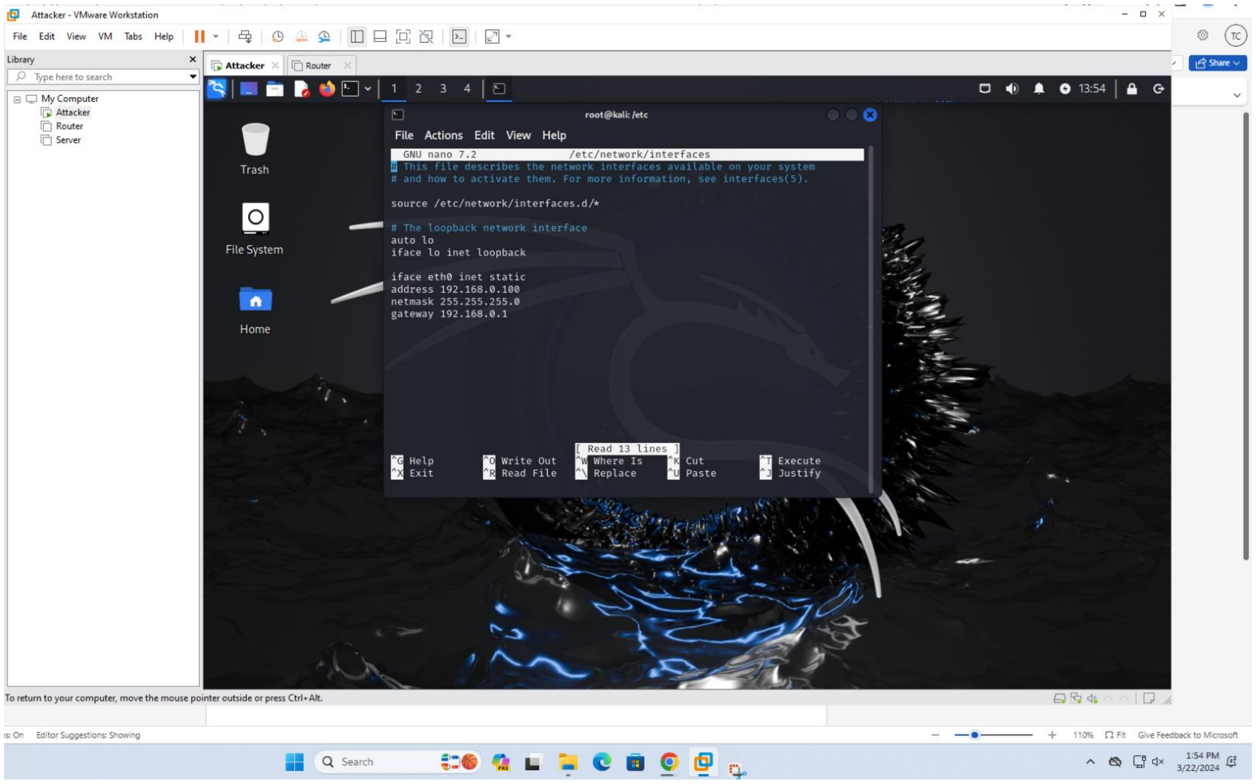
4 May 2024

Individual Project 2 - HTTP Attacks and Exploring HTTPS With Python

1. Setup a virtual environment (VMvare or VirtualBox), this has been done in Project 1
    - Three VMs: Attacker, Server, Router
    - Attacker and Server are in two different networks
    - The three machines are able to ping each other
    - Able to use Wireshark or Tcpdump to capture pass-through traffic

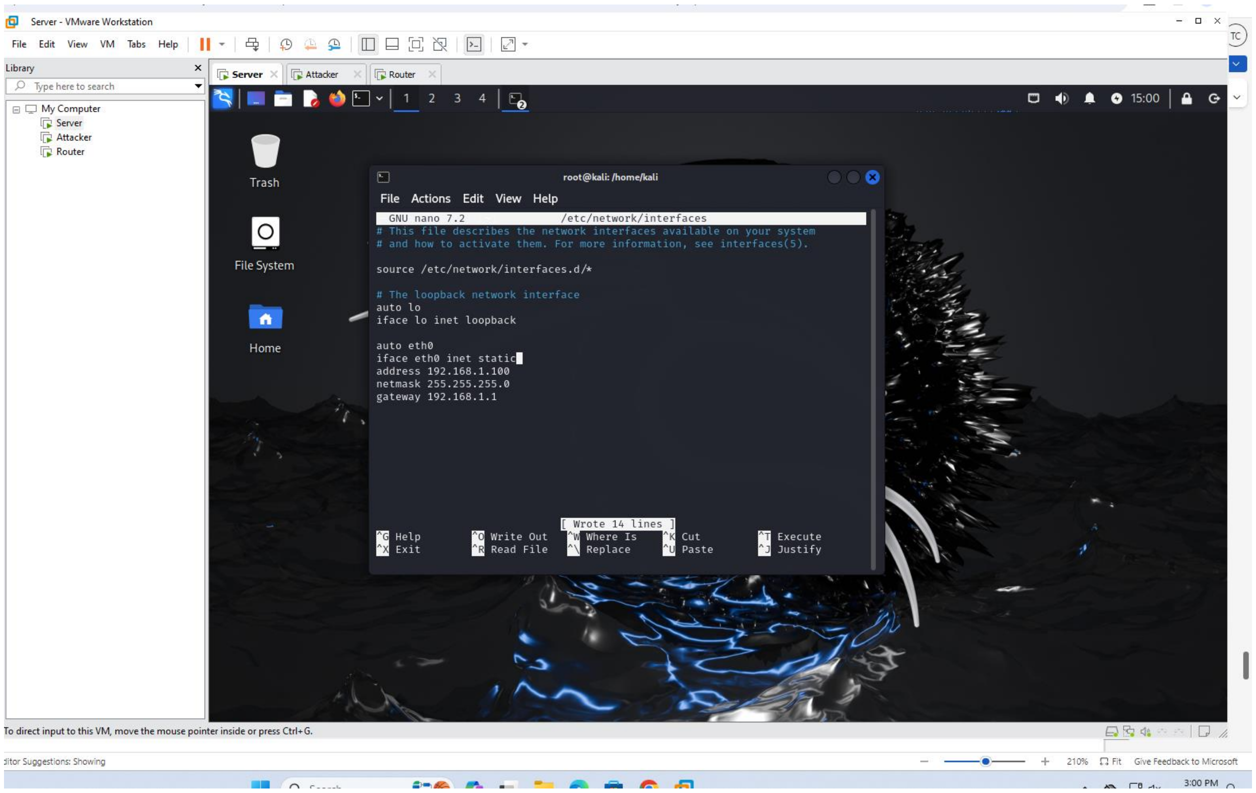
I used the virtual environment created in project 1. Here are the screenshots for the interfaces for each again.

Attacker machine


Attacker - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Attacker
Router
Server
Trash
File System
Home
root@kali: /etc
File Actions Edit View Help
13:54
To return to your computer, move the mouse pointer outside or press Ctrl+Alt.
1:54 PM
3/22/2024

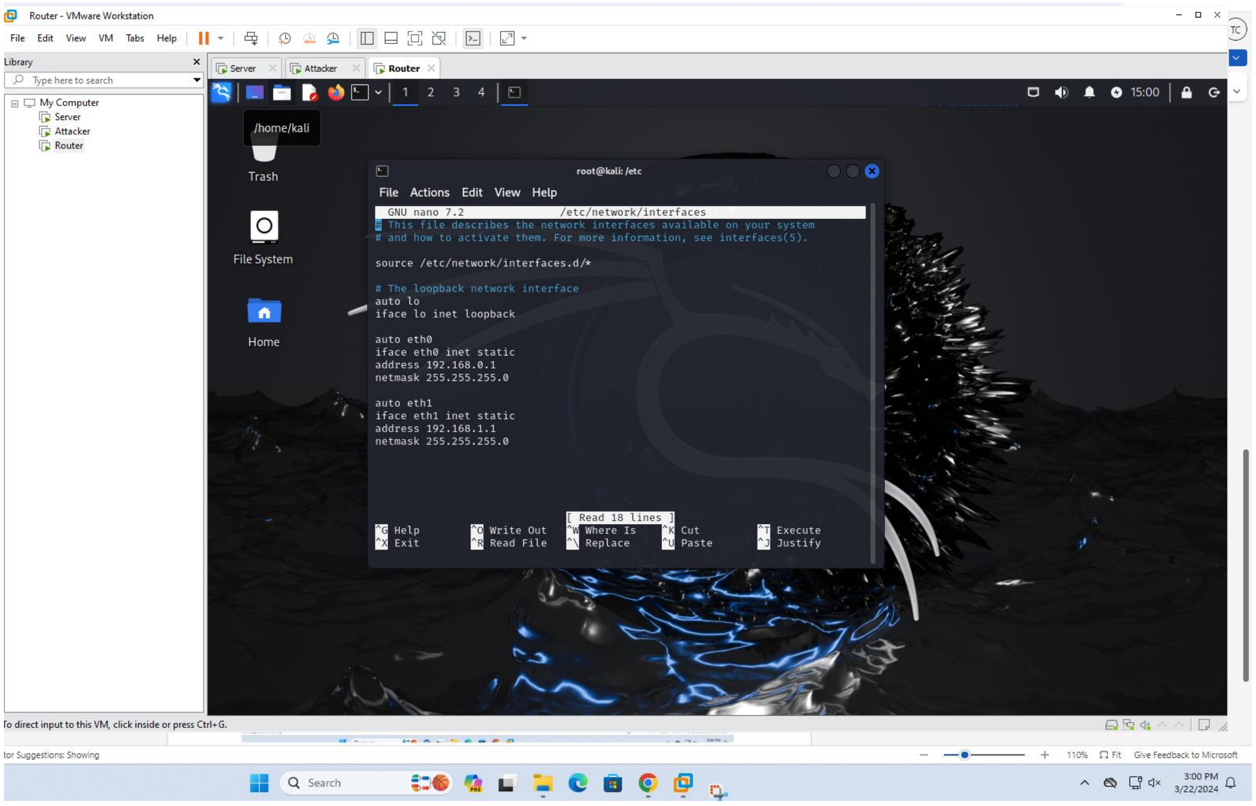
```
  GNU nano 7.2                 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

iface eth0 inet static
address 192.168.0.100
netmask 255.255.255.0
gateway 192.168.0.1

                              [ Read 13 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Server machine



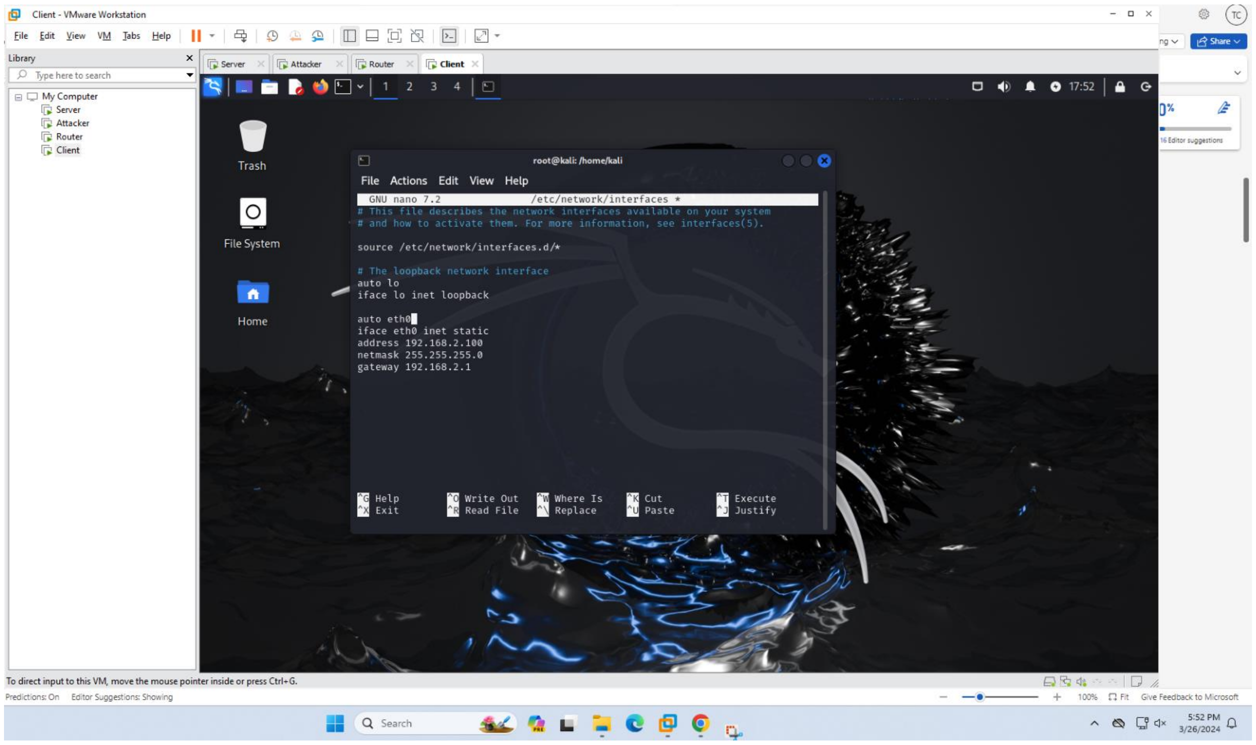
```
  GNU nano 7.2                /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.100
netmask 255.255.255.0
gateway 192.168.1.1

                              [ Wrote 14 lines ]
^G Help       ^O Write Out   ^W Where Is    ^K Cut        ^T Execute
^X Exit       ^R Read File   ^\ Replace     ^U Paste      ^J Justify
```

Router machine



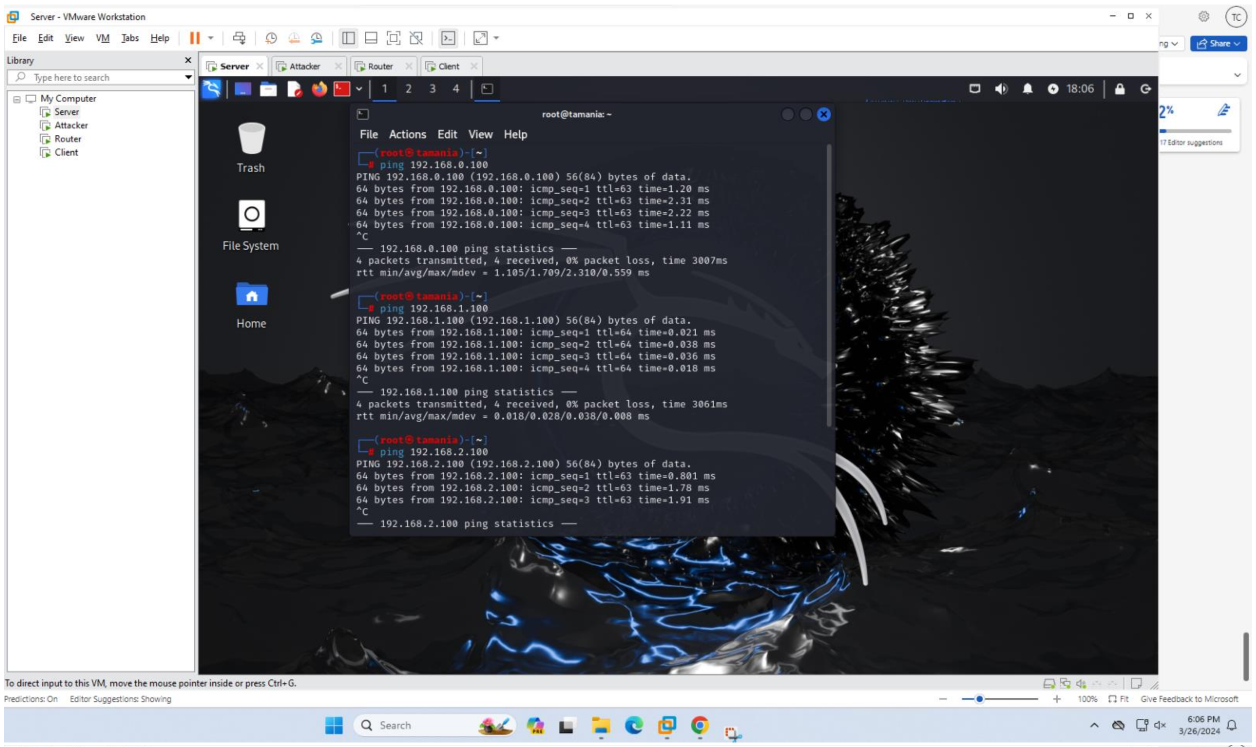
```
  GNU nano 7.2                /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0

auto eth1
iface eth1 inet static
address 192.168.1.1
netmask 255.255.255.0

                              [ Read 18 lines ]
^G Help       ^O Write Out   ^W Where Is    ^K Cut        ^T Execute
^X Exit       ^R Read File   ^\ Replace     ^U Paste      ^J Justify
```

Client machine

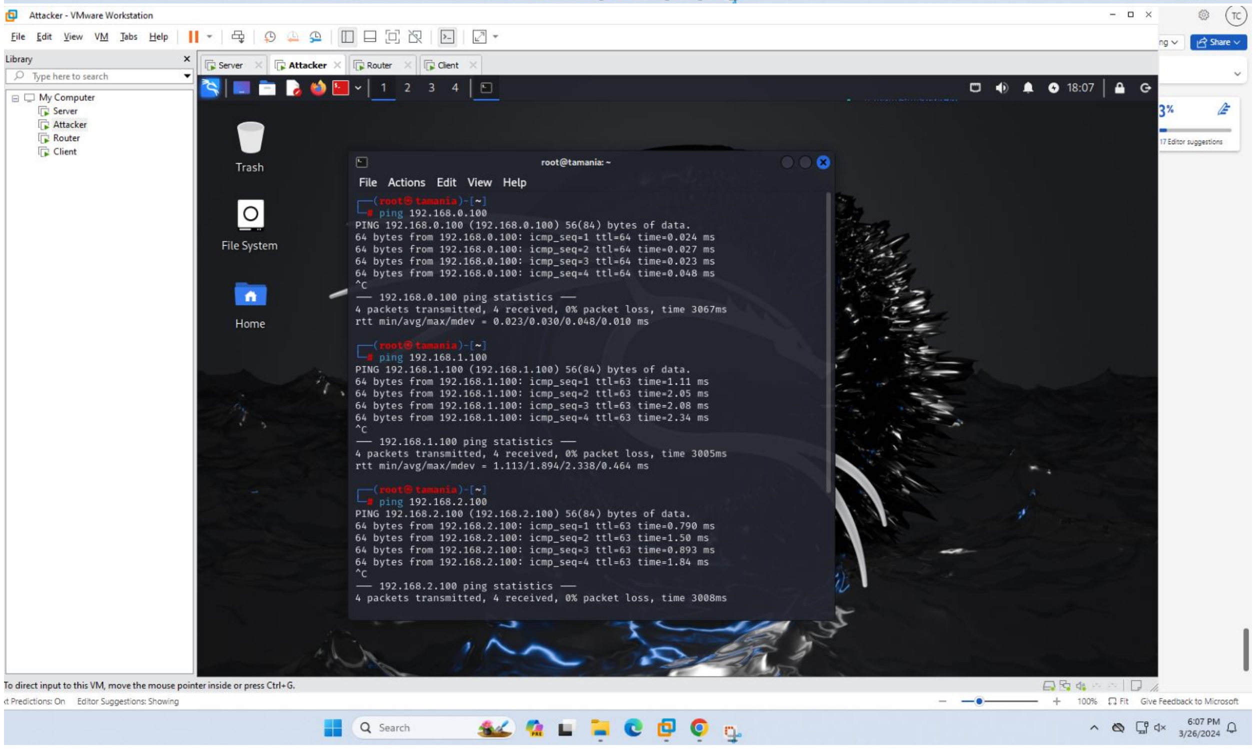


```
  GNU nano 7.2                /etc/network/interfaces *
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.2.100
netmask 255.255.255.0
gateway 192.168.2.1
```

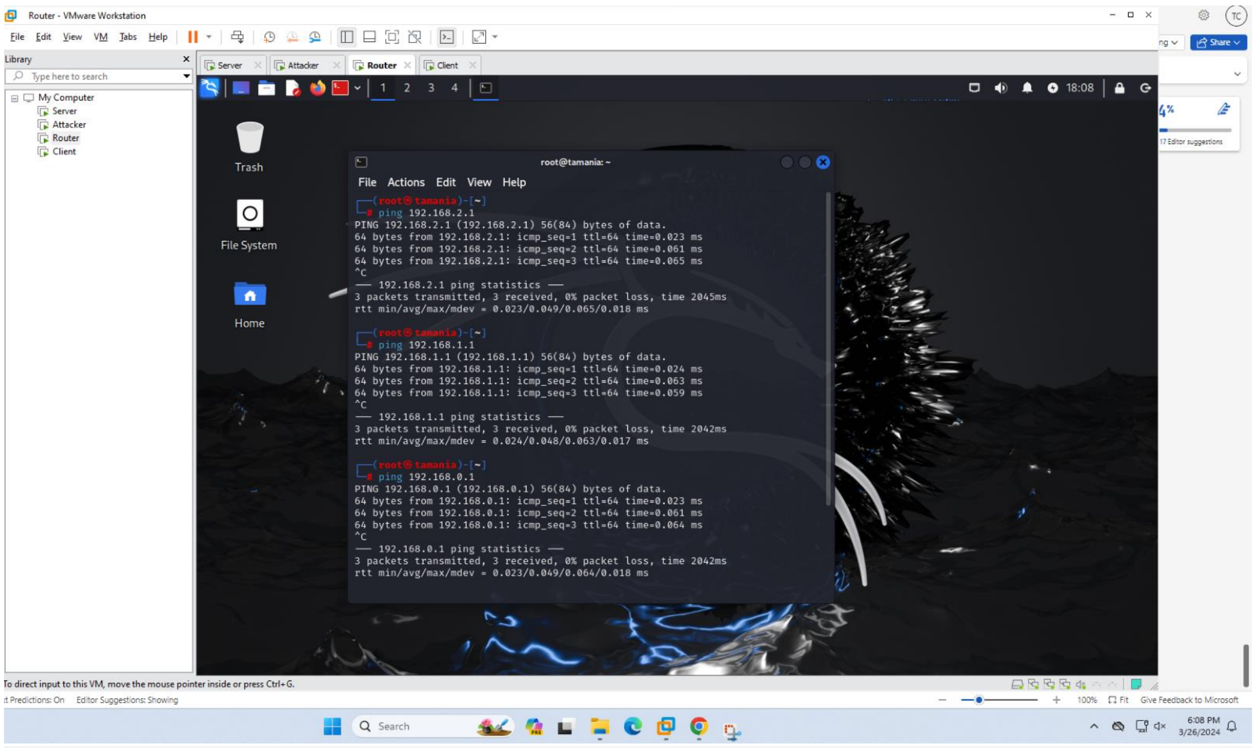
The machines can ping each other.


Server - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Server
Attacker
Router
Client
Trash
File System
Home
18:06
root@tamania: ~
File Actions Edit View Help
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
6:06 PM
3/26/2024

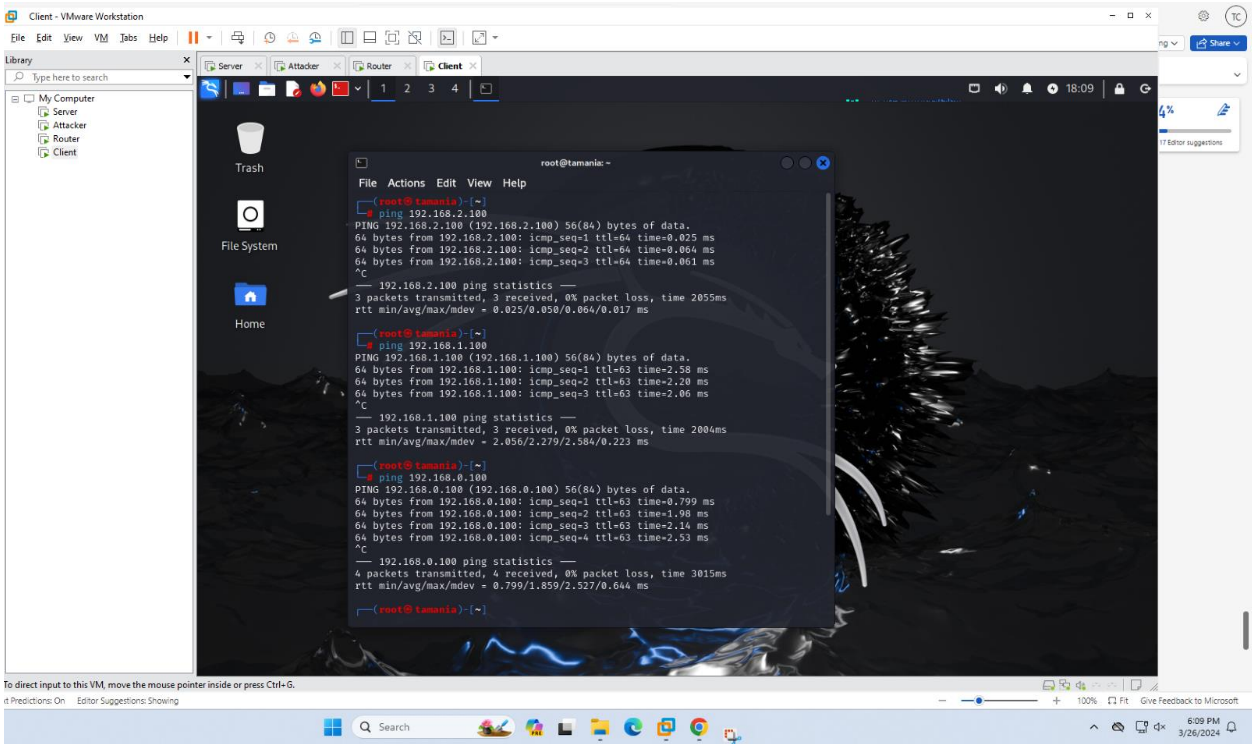
```
┌──(root㉿tamania)-[~]
└─# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=1.20 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=2.31 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=63 time=2.22 ms
64 bytes from 192.168.0.100: icmp_seq=4 ttl=63 time=1.11 ms
^C
── 192.168.0.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 1.105/1.709/2.310/0.559 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=64 time=0.021 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=64 time=0.018 ms
^C
── 192.168.1.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.018/0.028/0.038/0.008 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
64 bytes from 192.168.2.100: icmp_seq=1 ttl=63 time=0.801 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=63 time=1.78 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=63 time=1.91 ms
^C
── 192.168.2.100 ping statistics ──
```


Attacker - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Server
Attacker
Router
Client
Trash
File System
Home
18:07
root@tamania: ~
File Actions Edit View Help
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
6:07 PM
3/26/2024

```
┌──(root㉿tamania)-[~]
└─# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=64 time=0.027 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=64 time=0.023 ms
64 bytes from 192.168.0.100: icmp_seq=4 ttl=64 time=0.048 ms
^C
── 192.168.0.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.023/0.030/0.048/0.010 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=63 time=1.11 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=63 time=2.05 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=63 time=2.08 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=63 time=2.34 ms
^C
── 192.168.1.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.113/1.894/2.338/0.464 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
64 bytes from 192.168.2.100: icmp_seq=1 ttl=63 time=0.790 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=63 time=1.50 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=63 time=0.893 ms
64 bytes from 192.168.2.100: icmp_seq=4 ttl=63 time=1.84 ms
^C
── 192.168.2.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
```


Router - VMware Workstation
Server
Attacker
Router
Client
Trash
File System
Home
root@tamania: ~
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```
┌──(root㉿tamania)-[~]
└─# ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=64 time=0.065 ms
^C
── 192.168.2.1 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.023/0.049/0.065/0.018 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.059 ms
^C
── 192.168.1.1 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.024/0.048/0.063/0.017 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.064 ms
^C
── 192.168.0.1 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.023/0.049/0.064/0.018 ms
```

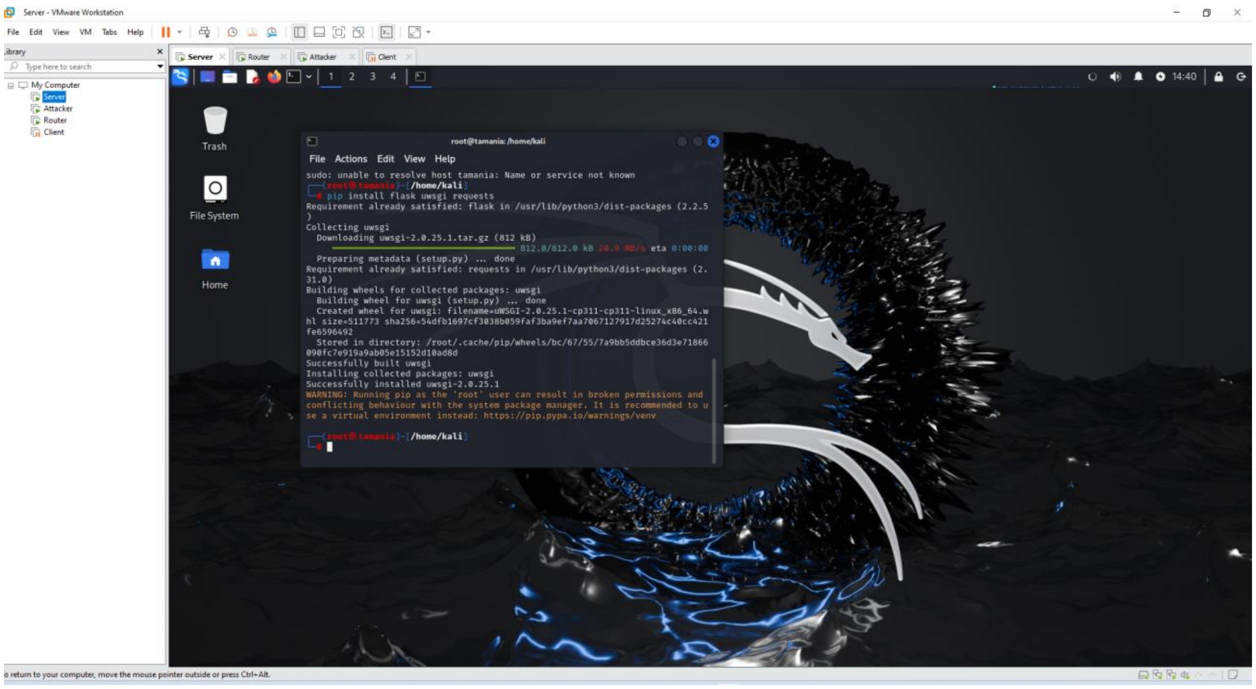

Client - VMware Workstation
Server
Attacker
Router
Client
Trash
File System
Home
root@tamania: ~
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```
┌──(root㉿tamania)-[~]
└─# ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
64 bytes from 192.168.2.100: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=64 time=0.061 ms
^C
── 192.168.2.100 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.025/0.050/0.064/0.017 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=63 time=2.58 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=63 time=2.20 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=63 time=2.06 ms
^C
── 192.168.1.100 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 2.056/2.279/2.584/0.223 ms

┌──(root㉿tamania)-[~]
└─# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=0.799 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=1.98 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=63 time=2.14 ms
64 bytes from 192.168.0.100: icmp_seq=4 ttl=63 time=2.53 ms
^C
── 192.168.0.100 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3015ms
rtt min/avg/max/mdev = 0.799/1.859/2.527/0.644 ms

┌──(root㉿tamania)-[~]
```

2. Creating an Example Application

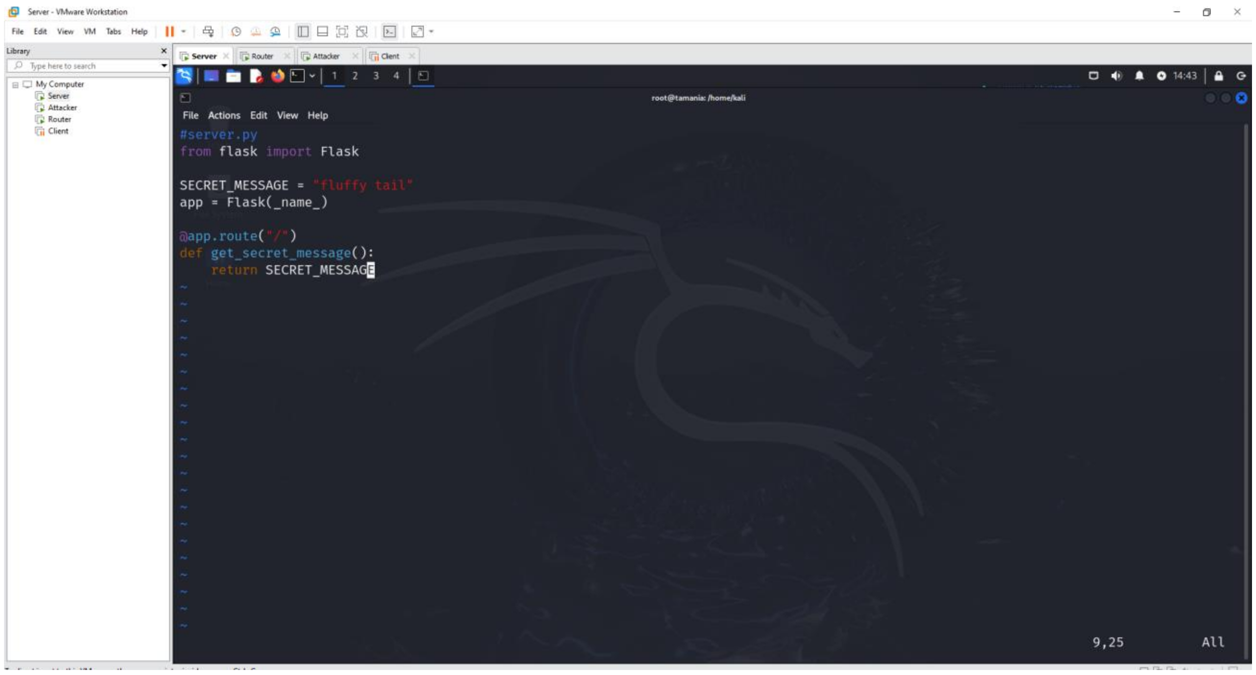
- Flask to build a web application
- uWSGI as a production server
- requests to exercise your server

Downloading uwsgi

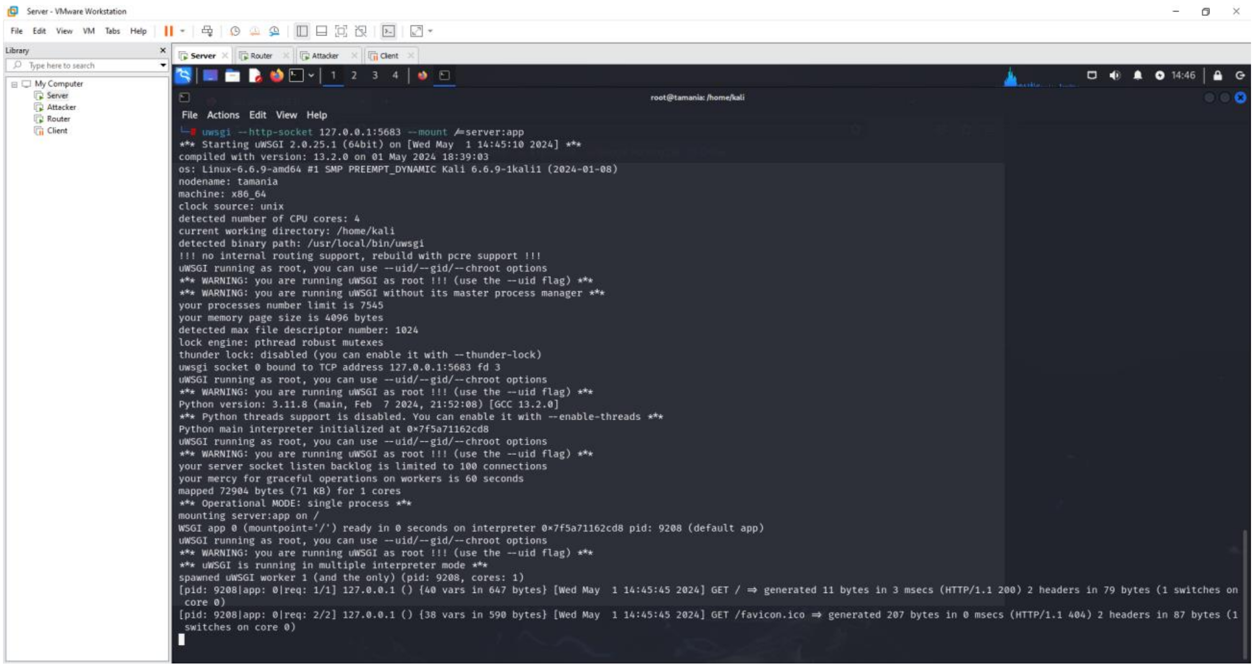


```
sudo: unable to resolve host tamania: Name or service not known
┌──(root㉿tamania)-[/home/kali]
└─# pip install flask uwsgi requests
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (2.2.5
)
Collecting uwsgi
  Downloading uwsgi-2.0.25.1.tar.gz (812 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 812.0/812.0 kB 20.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.
31.0)
Building wheels for collected packages: uwsgi
  Building wheel for uwsgi (setup.py) ... done
  Created wheel for uwsgi: filename=uWSGI-2.0.25.1-cp311-cp311-linux_x86_64.w
hl size=511773 sha256=54dfb1697cf3038b059faf3ba9ef7aa7067127917d25274c40cc421
fe6596492
  Stored in directory: /root/.cache/pip/wheels/bc/67/55/7a9bb5ddbce36d3e71866
090fc7e919a9ab05e15152d10ad8d
Successfully built uwsgi
Installing collected packages: uwsgi
Successfully installed uwsgi-2.0.25.1
WARNING: Running pip as the 'root' user can result in broken permissions and
conflicting behaviour with the system package manager. It is recommended to u
se a virtual environment instead: https://pip.pypa.io/warnings/venv

┌──(root㉿tamania)-[/home/kali]
└─# 
```

Create flask application

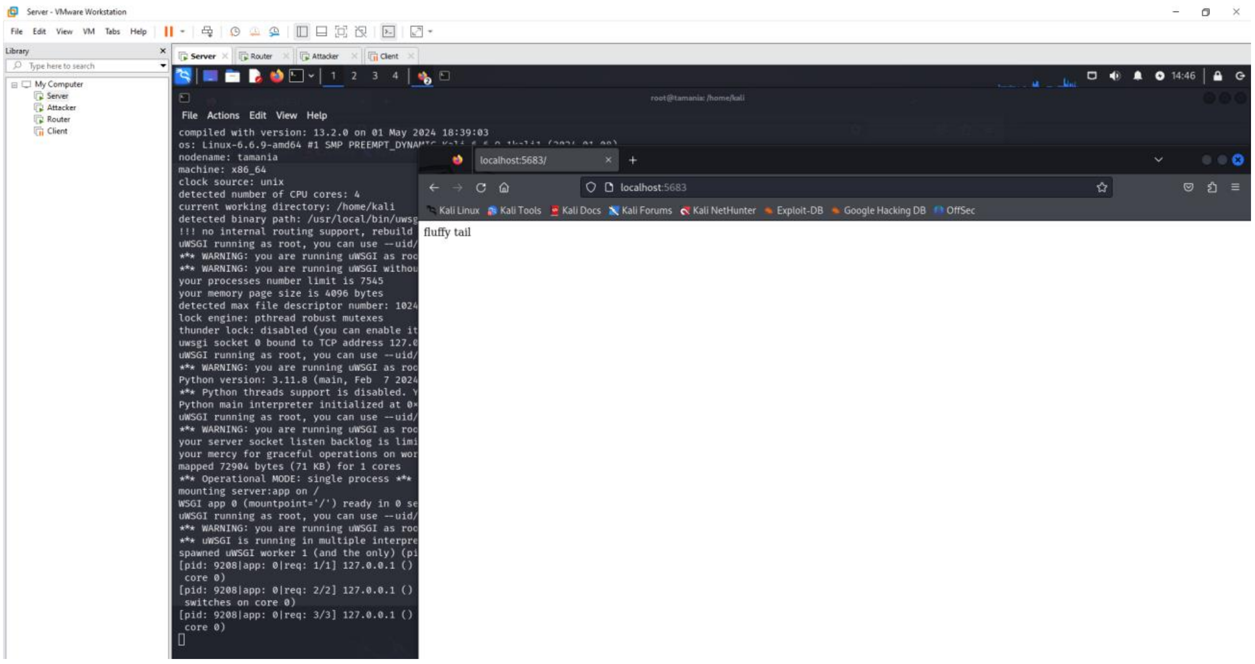


```
#server.py
from flask import Flask

SECRET_MESSAGE = "fluffy tail"
app = Flask(_name_)

@app.route("/")
def get_secret_message():
    return SECRET_MESSAGE
```



```
└─# uwsgi --http-socket 127.0.0.1:5683 --mount /=server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Wed May  1 14:45:10 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 127.0.0.1:5683 fd 3
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Python version: 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x7f5a71162cd8
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 72904 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
mounting server:app on /
WSGI app 0 (mountpoint='/') ready in 0 seconds on interpreter 0x7f5a71162cd8 pid: 9208 (default app)
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 9208, cores: 1)
[pid: 9208|app: 0|req: 1/1] 127.0.0.1 () {40 vars in 647 bytes} [Wed May  1 14:45:45 2024] GET / ⇒ generated 11 bytes in 3 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on
 core 0)
[pid: 9208|app: 0|req: 2/2] 127.0.0.1 () {38 vars in 590 bytes} [Wed May  1 14:45:45 2024] GET /favicon.ico ⇒ generated 207 bytes in 0 msecs (HTTP/1.1 404) 2 headers in 87 bytes (1
 switches on core 0)
```

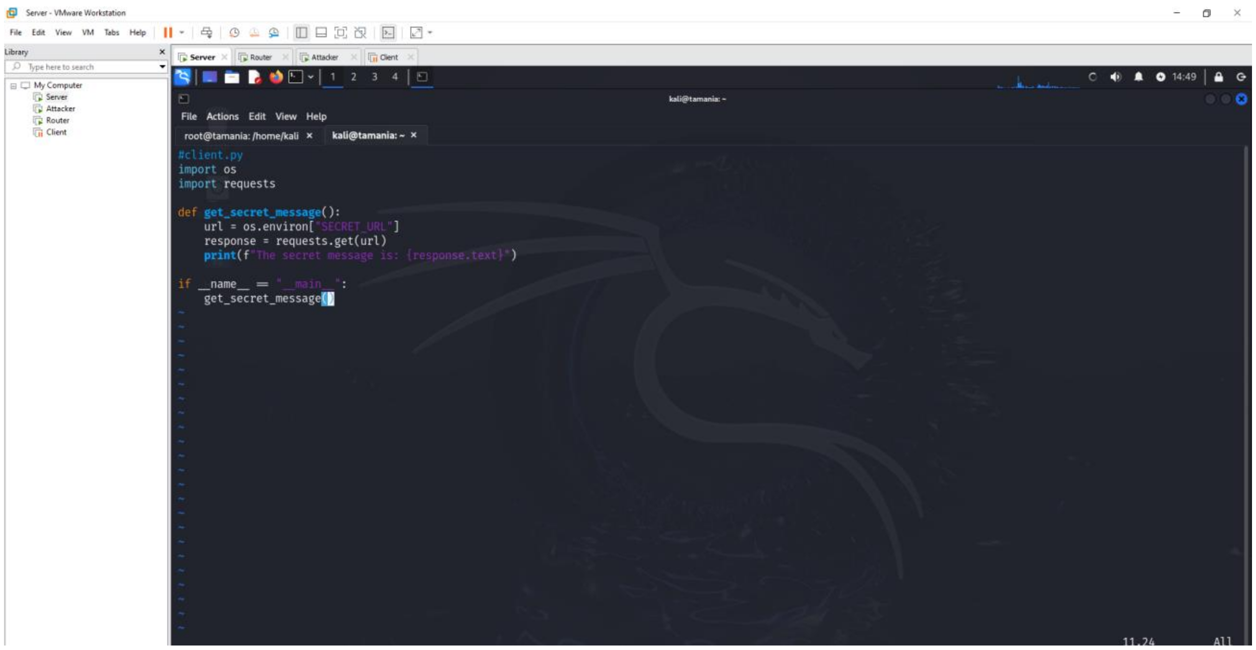
Confirm its working



```
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAM
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsg
!!! no internal routing support, rebuild
uWSGI running as root, you can use --uid/
*** WARNING: you are running uWSGI as roo
*** WARNING: you are running uWSGI withou
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it
uwsgi socket 0 bound to TCP address 127.0
uWSGI running as root, you can use --uid/
*** WARNING: you are running uWSGI as roo
Python version: 3.11.8 (main, Feb  7 2024
*** Python threads support is disabled. Y
Python main interpreter initialized at 0x
uWSGI running as root, you can use --uid/
*** WARNING: you are running uWSGI as roo
your server socket listen backlog is limi
your mercy for graceful operations on wor
mapped 72904 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
mounting server:app on /
WSGI app 0 (mountpoint='/') ready in 0 se
uWSGI running as root, you can use --uid/
*** WARNING: you are running uWSGI as roo
*** uWSGI is running in multiple interpre
spawned uWSGI worker 1 (and the only) (pi
[pid: 9208|app: 0|req: 1/1] 127.0.0.1 ()
 core 0)
[pid: 9208|app: 0|req: 2/2] 127.0.0.1 ()
 switches on core 0)
[pid: 9208|app: 0|req: 3/3] 127.0.0.1 ()
 core 0)
```

fluffy tail

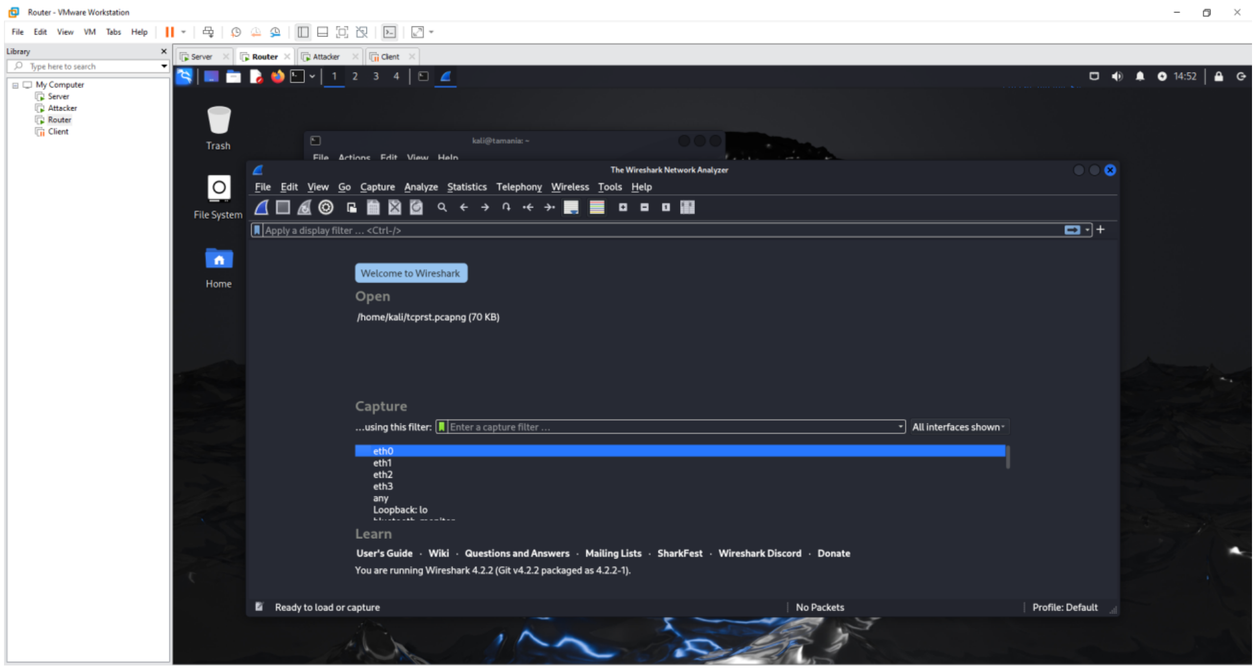
Client.py



```
#client.py
import os
import requests

def get_secret_message():
    url = os.environ["SECRET_URL"]
    response = requests.get(url)
    print(f"The secret message is: {response.text}")

if __name__ == "__main__":
    get_secret_message()
```

3. (20 pts) Use wireshark to show HTTP is not safe
    - Wireshark setup
    - Run experiments to capture unsafe traffic

Wireshark installed on router



Welcome to Wireshark

Open

/home/kali/tcprst.pcapng (70 KB)

Capture

...using this filter: Enter a capture filter ... All interfaces shown

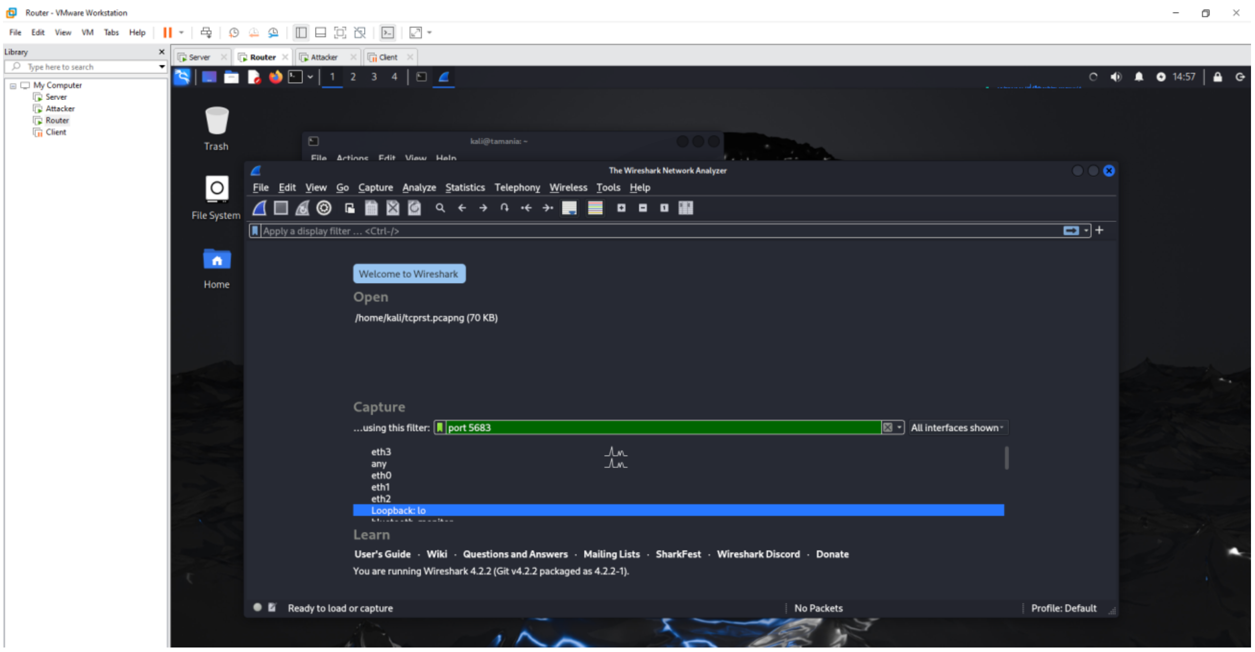
```
eth0
eth1
eth2
eth3
any
Loopback: lo
```

Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

You are running Wireshark 4.2.2 (Git v4.2.2 packaged as 4.2.2-1).

Ready to load or capture | No Packets | Profile: Default



/home/kali/tcprst.pcapng (70 KB)

...using this filter: port 5683

All interfaces shown

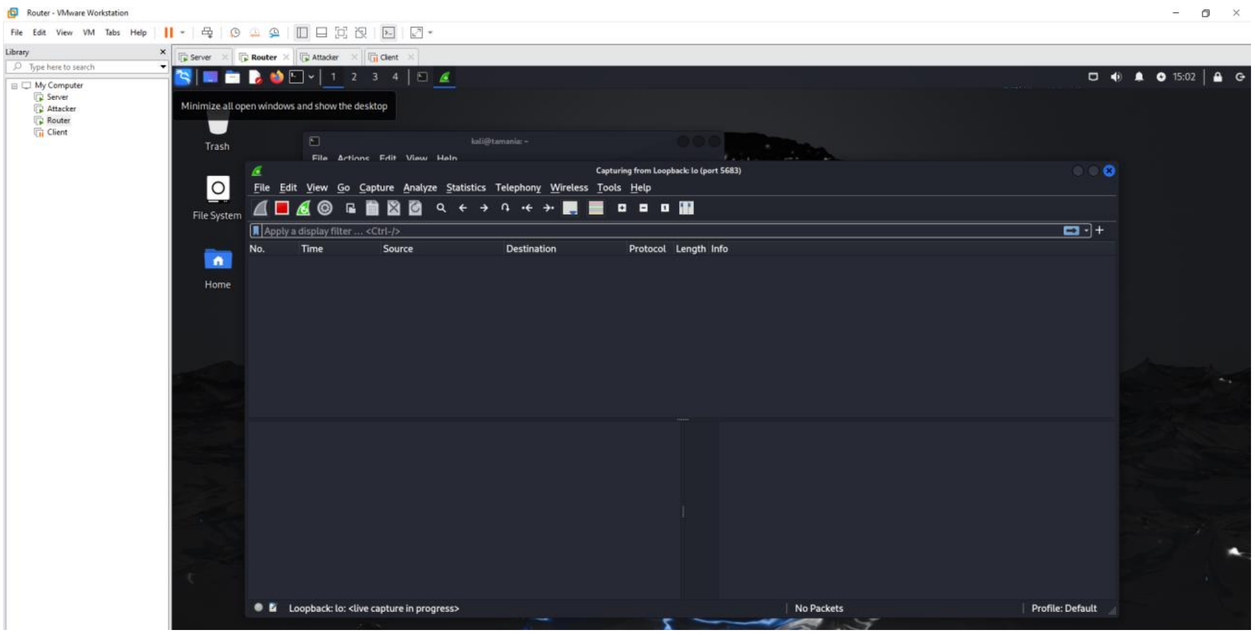
eth3
any
eth0
eth1
eth2
Loopback: lo

User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

You are running Wireshark 4.2.2 (Git v4.2.2 packaged as 4.2.2-1).

Ready to load or capture | No Packets | Profile: Default

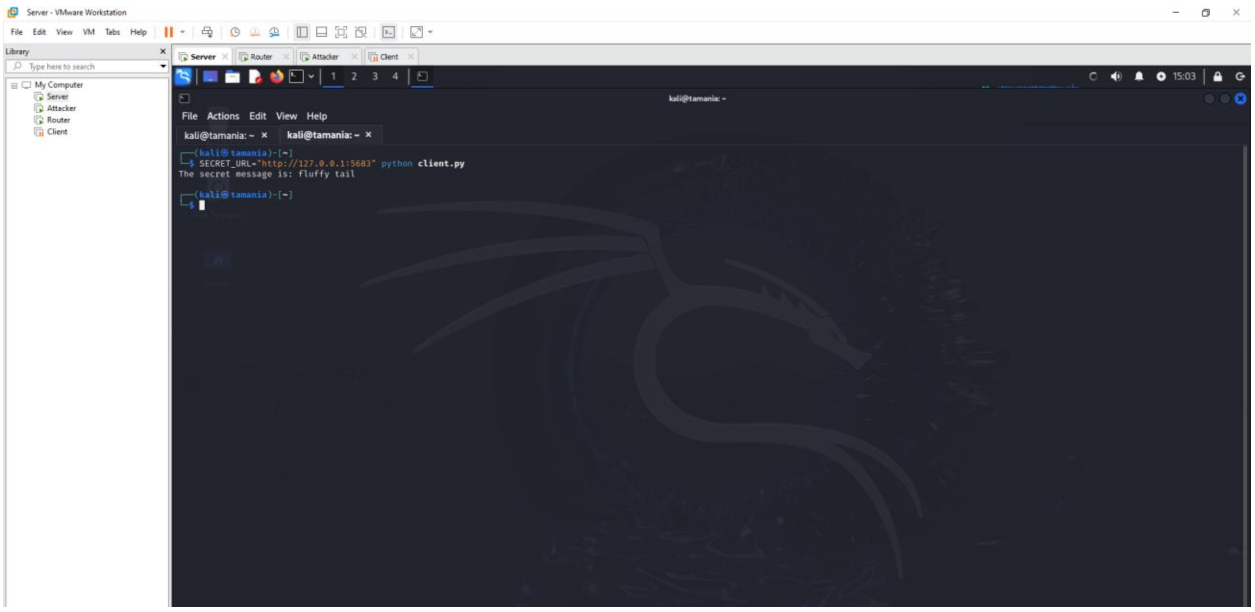
Started wireshark for port 5863


Router - VMware Workstation
Server
Attacker
Router
Client
Minimize all open windows and show the desktop
Capturing from Loopback: lo (port 5683)

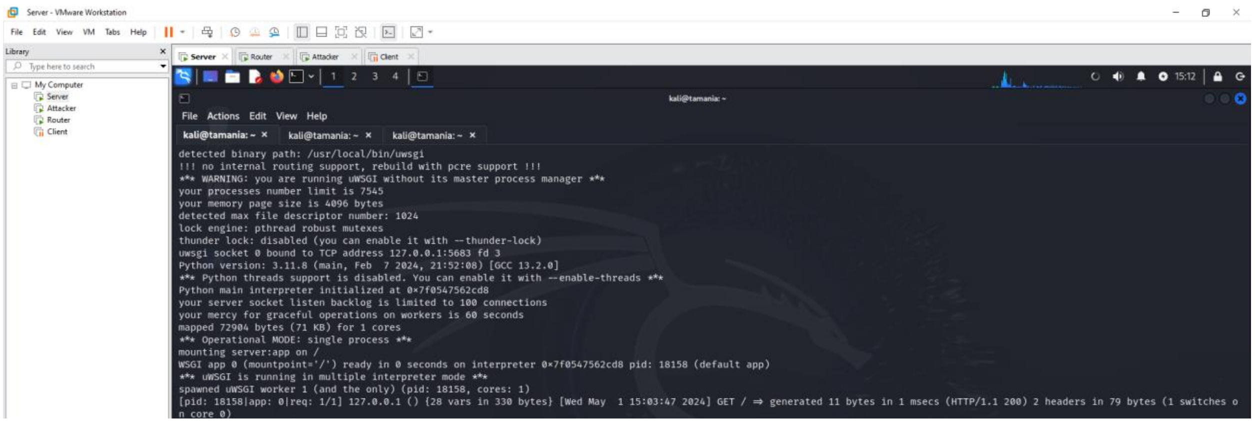
No.	Time	Source	Destination	Protocol	Length	Info

Loopback: lo: <live capture in progress> | No Packets | Profile: Default

Get some data

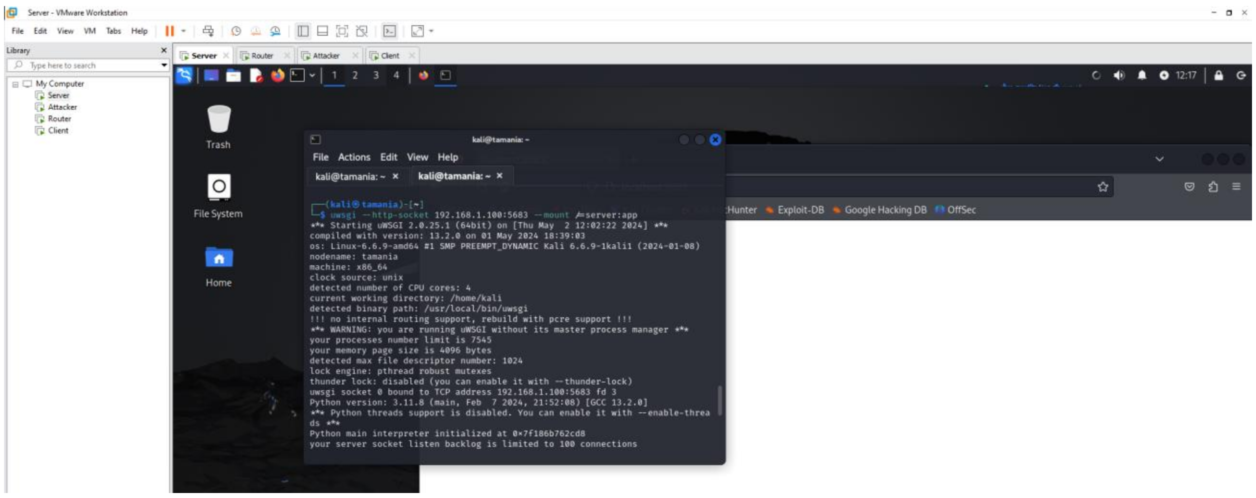


```
┌──(kali㉿tamania)-[~]
└─$ SECRET_URL="http://127.0.0.1:5683" python client.py
The secret message is: fluffy tail

┌──(kali㉿tamania)-[~]
└─$ 
```



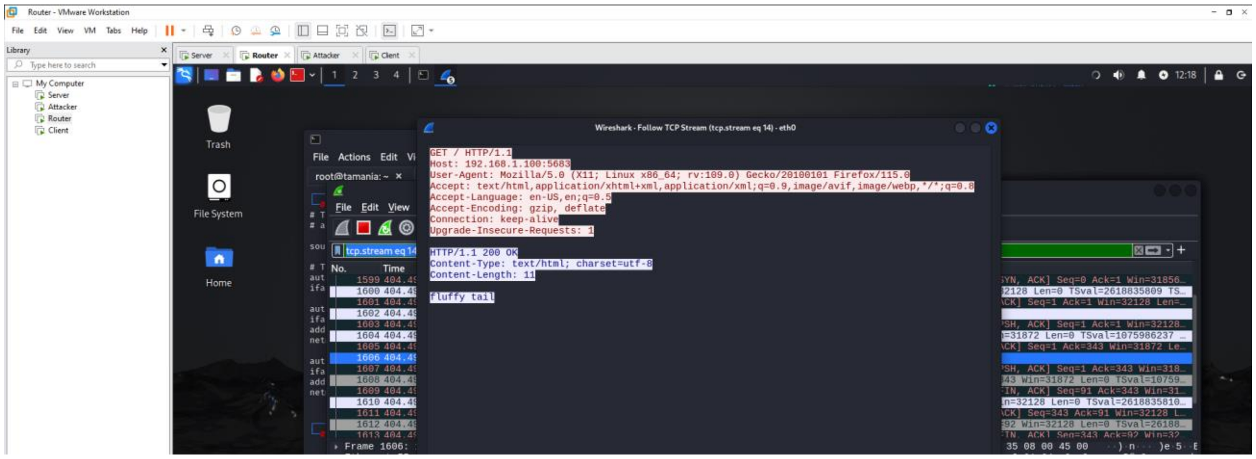
```
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 127.0.0.1:5683 fd 3
Python version: 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x7f0547562cd8
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 72904 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
mounting server:app on /
WSGI app 0 (mountpoint='/') ready in 0 seconds on interpreter 0x7f0547562cd8 pid: 18158 (default app)
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 18158, cores: 1)
[pid: 18158|app: 0|req: 1/1] 127.0.0.1 () {28 vars in 330 bytes} [Wed May  1 15:03:47 2024] GET / => generated 11 bytes in 1 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches o
n core 0)
```

In class, figured out wireshark wasn't working because mount line needed machine ip instead of 127.0.0.1



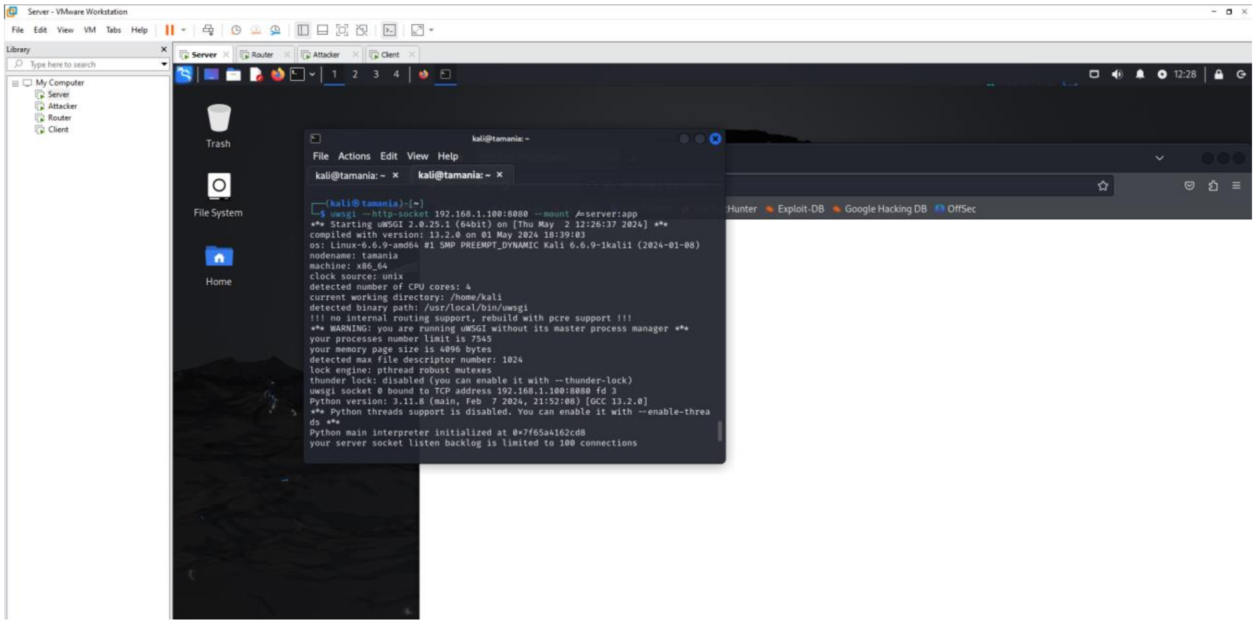
```
┌──(kali㉿tamania)-[~]
└─$ uwsgi --http-socket 192.168.1.100:5683 --mount /=server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Thu May  2 12:02:22 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 192.168.1.100:5683 fd 3
Python version: 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0]
*** Python threads support is disabled. You can enable it with --enable-threa
ds ***
Python main interpreter initialized at 0x7f186b762cd8
your server socket listen backlog is limited to 100 connections
```

From ack packet, shows message (uses Constrained Application Protocol).



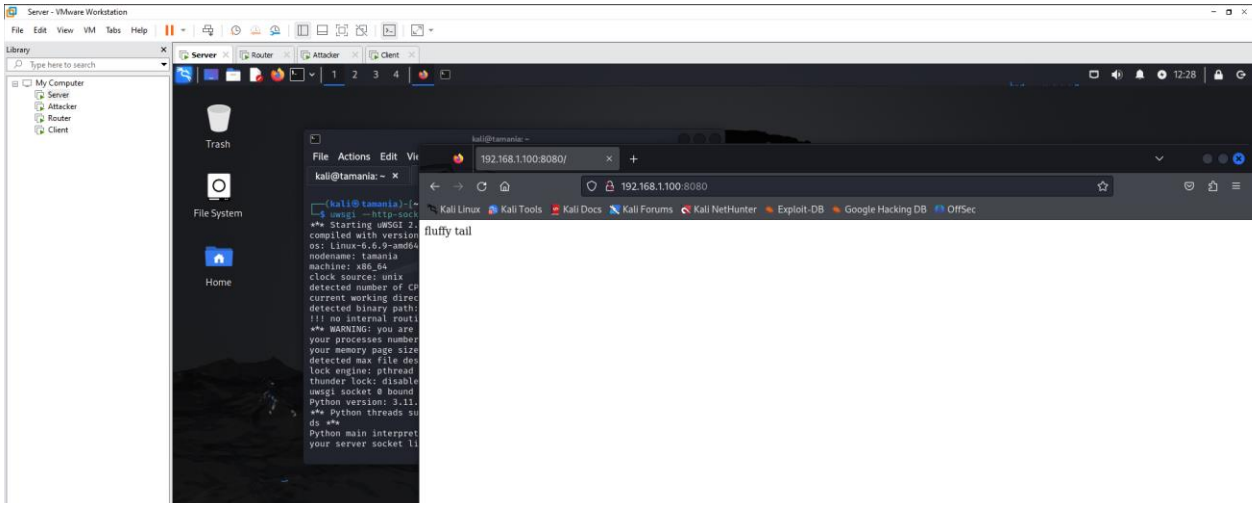
```
GET / HTTP/1.1
Host: 192.168.1.100:5683
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 11

fluffy tail
```

Later, we figured out that port 5683 may be the issue. Tried to do mount with port 8080 instead and it works.



```
┌──(kali㉿tamania)-[~]
└─$ uwsgi --http-socket 192.168.1.100:8080 --mount /=server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Thu May  2 12:26:37 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 192.168.1.100:8080 fd 3
Python version: 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0]
*** Python threads support is disabled. You can enable it with --enable-threa
ds ***
Python main interpreter initialized at 0x7f65a4162cd8
your server socket listen backlog is limited to 100 connections
```

On server machine

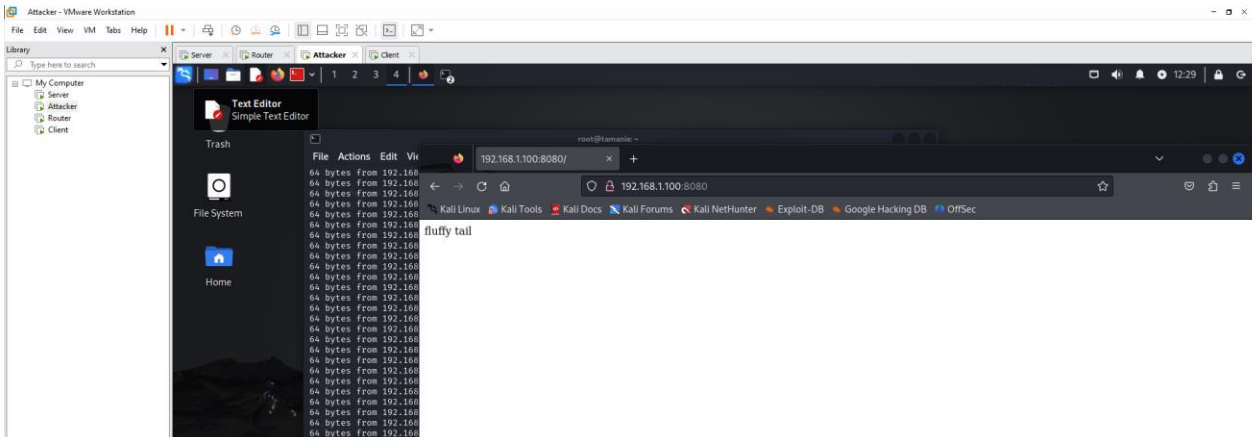


```
┌──(kali㉿tamania)-[~
└─$ uwsgi --http-sock
*** Starting uWSGI 2.
compiled with version
os: Linux-6.6.9-amd64
nodename: tamania
machine: x86_64
clock source: unix
detected number of CP
current working direc
detected binary path:
!!! no internal routi
*** WARNING: you are
your processes number
your memory page size
detected max file des
lock engine: pthread
thunder lock: disable
uwsgi socket 0 bound
Python version: 3.11.
*** Python threads su
ds ***
Python main interpret
your server socket li
```

192.168.1.100:8080

fluffy tail

On attacker machine

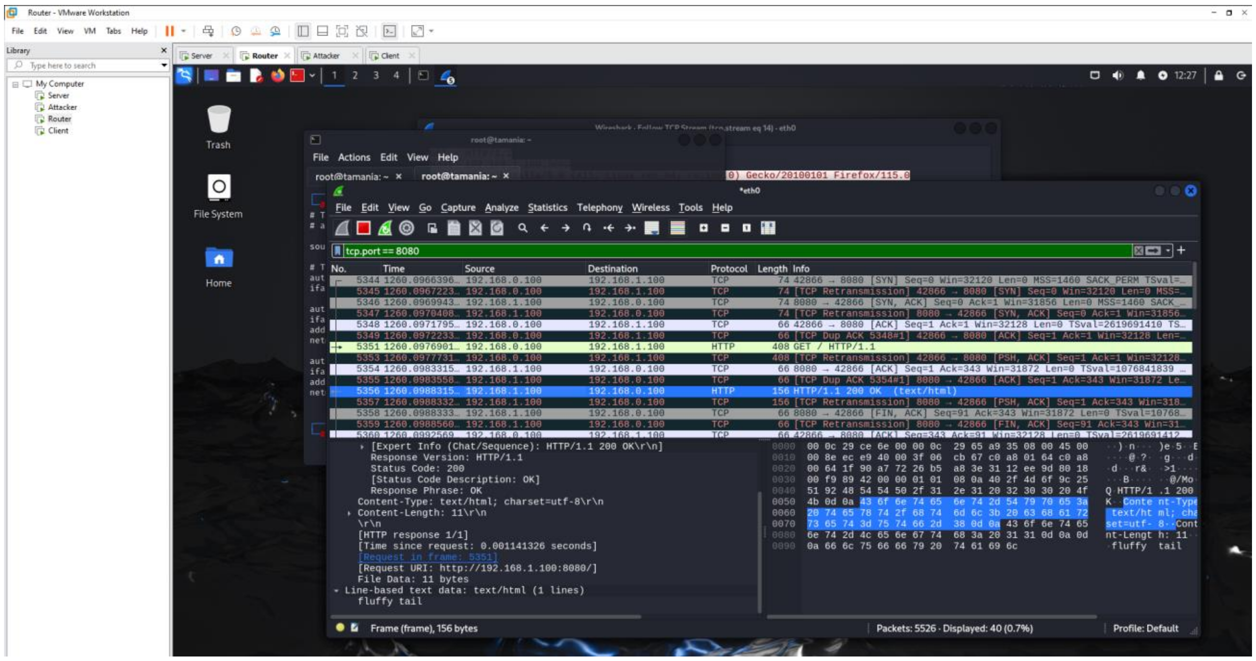


```
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
64 bytes from 192.168
```

192.168.1.100:8080

fluffy tail

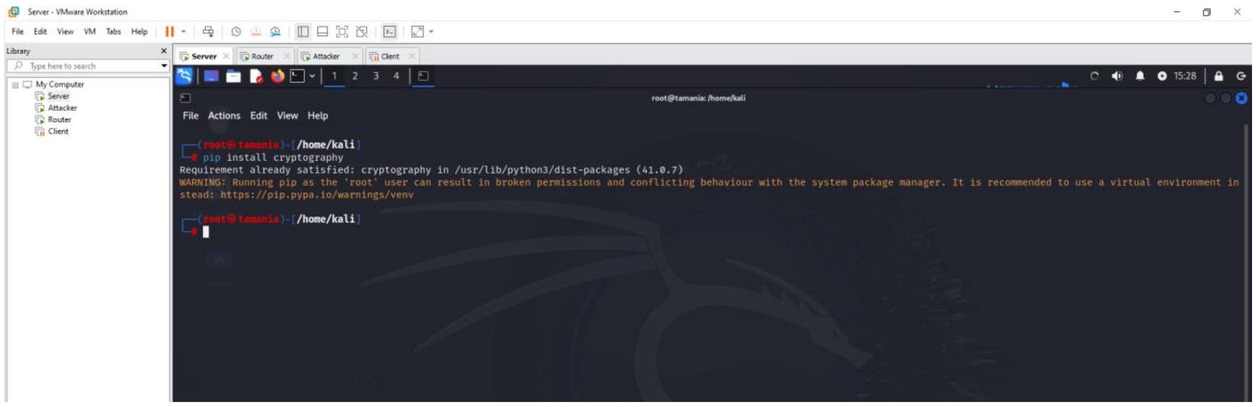
Wireshark captures secret message – fluffy tail



```
tcp.port == 8080
No.  Time           Source         Destination    Protocol Length Info
5344 1260.0966396… 192.168.0.100 192.168.1.100 TCP       74 42866 → 8080 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=…
5345 1260.0967223… 192.168.0.100 192.168.1.100 TCP       74 [TCP Retransmission] 42866 → 8080 [SYN] Seq=0 Win=32120 Len=0 MSS=…
5346 1260.0969943… 192.168.1.100 192.168.0.100 TCP       74 8080 → 42866 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_…
5347 1260.0970408… 192.168.1.100 192.168.0.100 TCP       74 [TCP Retransmission] 8080 → 42866 [SYN, ACK] Seq=0 Ack=1 Win=31856…
5348 1260.0971795… 192.168.0.100 192.168.1.100 TCP       66 42866 → 8080 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2619691410 TS…
5349 1260.0972233… 192.168.0.100 192.168.1.100 TCP       66 [TCP Dup ACK 5348#1] 42866 → 8080 [ACK] Seq=1 Ack=1 Win=32128 Len=…
5351 1260.0976901… 192.168.0.100 192.168.1.100 HTTP     408 GET / HTTP/1.1
5353 1260.0977731… 192.168.0.100 192.168.1.100 TCP      408 [TCP Retransmission] 42866 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=32128…
5354 1260.0983315… 192.168.1.100 192.168.0.100 TCP       66 8080 → 42866 [ACK] Seq=1 Ack=343 Win=31872 Len=0 TSval=1076841839 …
5355 1260.0983558… 192.168.1.100 192.168.0.100 TCP       66 [TCP Dup ACK 5354#1] 8080 → 42866 [ACK] Seq=1 Ack=343 Win=31872 Le…
5356 1260.0988315… 192.168.1.100 192.168.0.100 HTTP     156 HTTP/1.1 200 OK  (text/html)
5357 1260.0988332… 192.168.1.100 192.168.0.100 TCP      156 [TCP Retransmission] 8080 → 42866 [PSH, ACK] Seq=1 Ack=343 Win=318…
5358 1260.0988333… 192.168.1.100 192.168.0.100 TCP       66 8080 → 42866 [FIN, ACK] Seq=91 Ack=343 Win=31872 Len=0 TSval=10768…
5359 1260.0988560… 192.168.1.100 192.168.0.100 TCP       66 [TCP Retransmission] 8080 → 42866 [FIN, ACK] Seq=91 Ack=343 Win=31…
5360 1260.0992569… 192.168.0.100 192.168.1.100 TCP       66 42866 → 8080 [ACK] Seq=343 Ack=91 Win=32128 Len=0 TSval=2619691412

    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Content-Type: text/html; charset=utf-8\r\n
  Content-Length: 11\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.001141326 seconds]
  [Request in frame: 5351]
  [Request URI: http://192.168.1.100:8080/]
  File Data: 11 bytes
Line-based text data: text/html (1 lines)
  fluffy tail

0000  00 0c 29 ce 6e 00 00 0c  29 65 a9 35 08 00 45 00   ··)·n···)e·5··E·
0010  00 8e ec e9 40 00 3f 06  cb 67 c0 a8 01 64 c0 a8   ····@·?··g···d··
0020  00 64 1f 90 a7 72 26 b5  a8 3e 31 12 ee 9d 80 18   ·d···r&··>1·····
0030  00 f9 89 42 00 00 01 01  08 0a 40 2f 4d 6f 9c 25   ···B······@/Mo·%
0040  51 92 48 54 54 50 2f 31  2e 31 20 32 30 30 20 4f   Q·HTTP/1.1 200 O
0050  4b 0d 0a 43 6f 6e 74 65  6e 74 2d 54 79 70 65 3a   K··Content-Type:
0060  20 74 65 78 74 2f 68 74  6d 6c 3b 20 63 68 61 72    text/html; char
0070  73 65 74 3d 75 74 66 2d  38 0d 0a 43 6f 6e 74 65   set=utf-8··Conte
0080  6e 74 2d 4c 65 6e 67 74  68 3a 20 31 31 0d 0a 0d   nt-Length: 11···
0090  0a 66 6c 75 66 66 79 20  74 61 69 6c               ·fluffy tail

Frame (frame), 156 bytes        Packets: 5526 · Displayed: 40 (0.7%)        Profile: Default
```

4. **(30 pts) Use secret key to encrypt and decrypt traffic**
   - Use Fernet library for encryption and description
   - Use Wireshark to show the safe communication

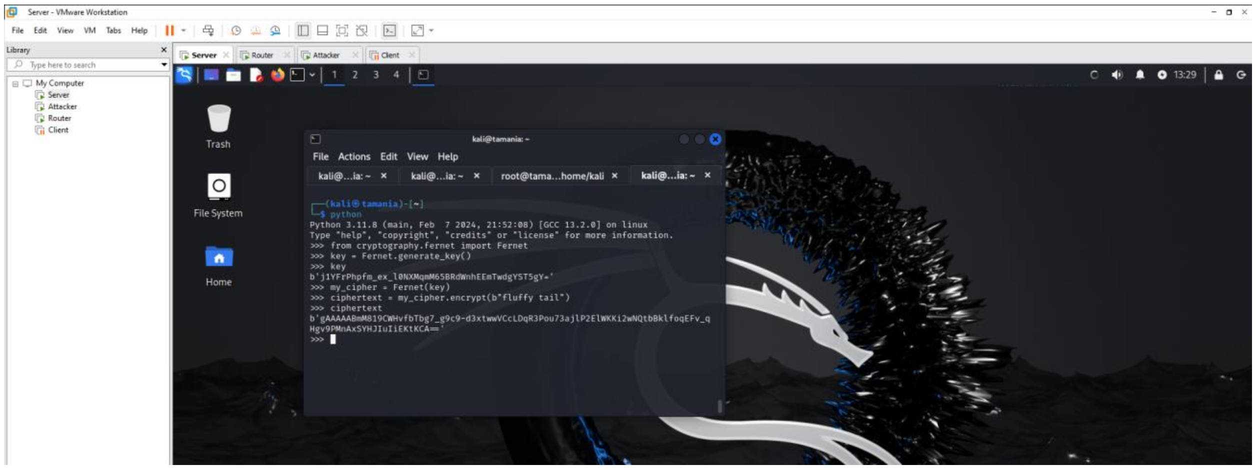
Install cryptography library



```
┌──(root㉿tamania)-[/home/kali]
└─# pip install cryptography
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (41.0.7)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment in
stead: https://pip.pypa.io/warnings/venv

┌──(root㉿tamania)-[/home/kali]
└─#
```

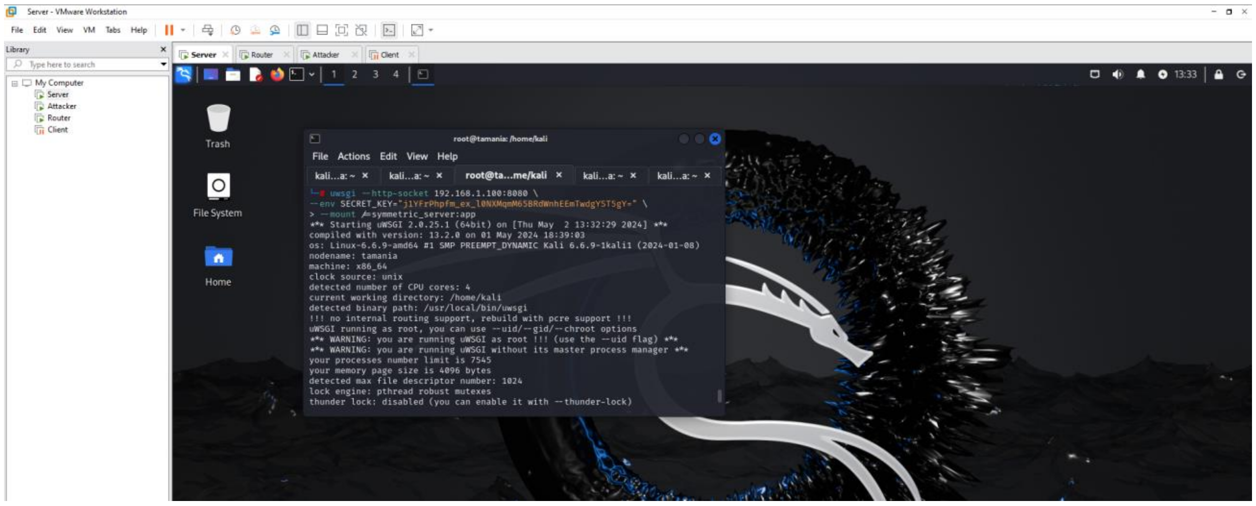
Generate key = b'j1YFrPhpfm_ex_l0NXMqmM65BRdWnhEEmTwdgYST5gY='

Encrypt message (fluffy tail) = b'gAAAAABmM819CWHvfbTbg7_g9c9-d3xtwwVCcLDqR3Pou73ajlP2ElWKKi2wNQtbBklfoqEFv_qHgv9PMnAxSYHJIuIiEKtKCA=='

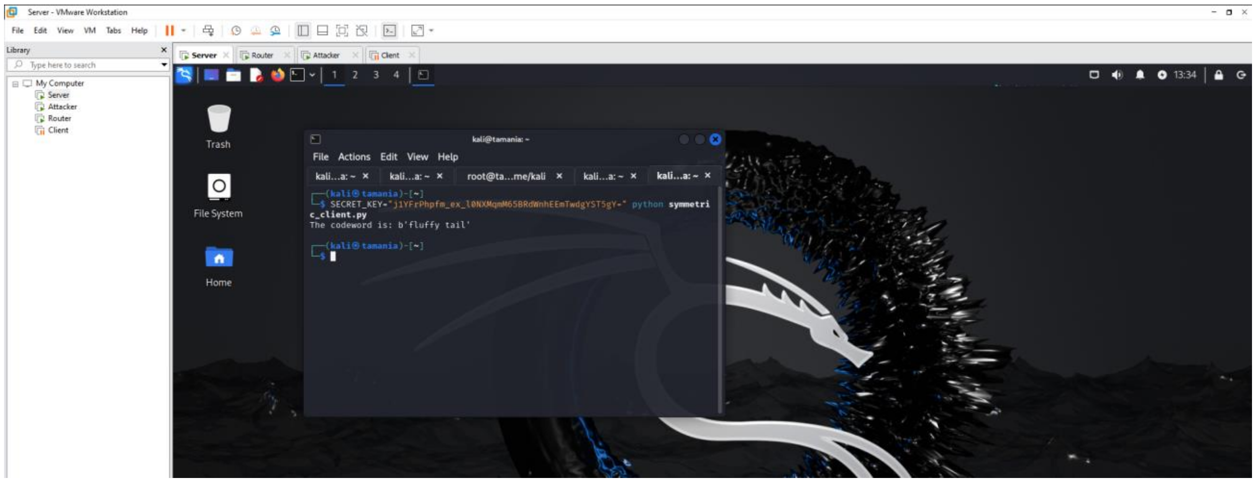


```
┌──(kali㉿tamania)-[~]
└─$ python
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cryptography.fernet import Fernet
>>> key = Fernet.generate_key()
>>> key
b'j1YFrPhpfm_ex_l0NXMqmM65BRdWnhEEmTwdgYST5gY='
>>> my_cipher = Fernet(key)
>>> ciphertext = my_cipher.encrypt(b"fluffy tail")
>>> ciphertext
b'gAAAAABmM819CWHvfbTbg7_g9c9-d3xtwwVCcLDqR3Pou73ajlP2ElWKKi2wNQtbBklfoqEFv_q
Hgv9PMnAxSYHJIuIiEKtKCA=='
>>> 
```

Run with new key

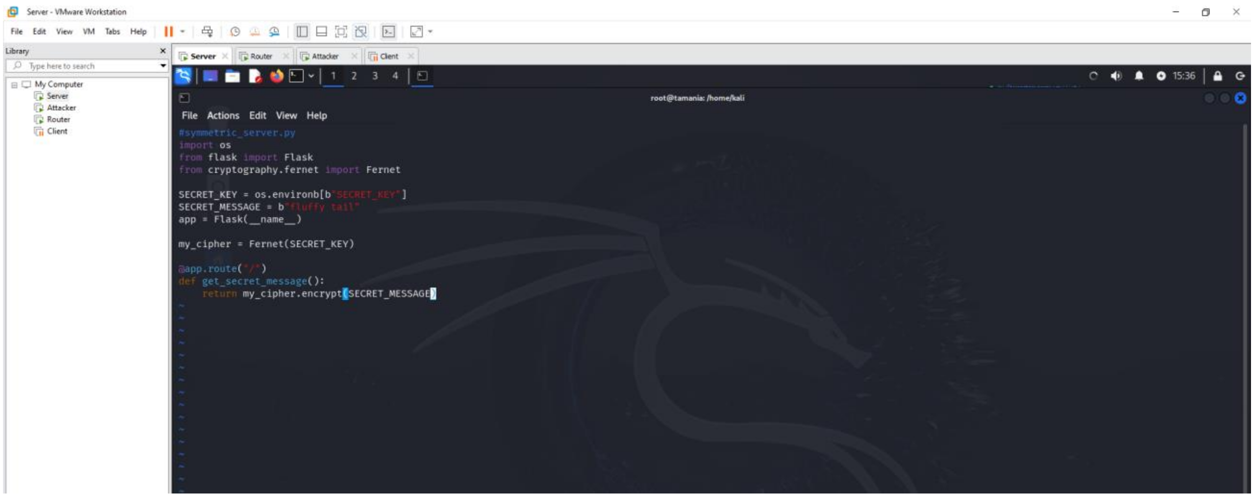


```
└─# uwsgi --http-socket 192.168.1.100:8080 \
--env SECRET_KEY="j1YFrPhpfm_ex_l0NXMqmM65BRdWnhEEmTwdgYST5gY=" \
> --mount /=symmetric_server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Thu May  2 13:32:29 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
```



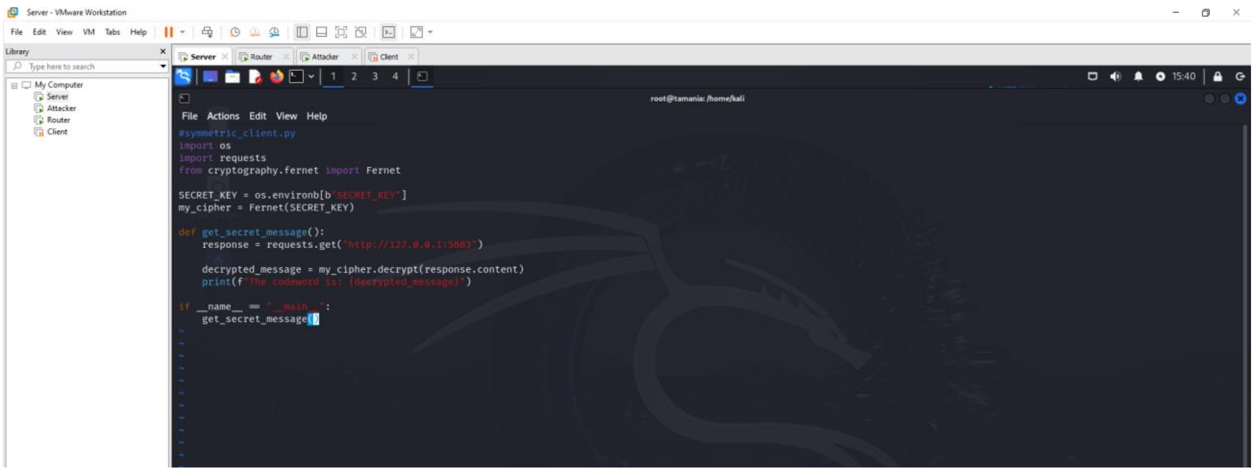
```
┌──(kali㉿tamania)-[~]
└─$ SECRET_KEY="j1YFrPhpfm_ex_l0NXMqmM65BRdWnhEEmTwdgYST5gY=" python symmetri
c_client.py
The codeword is: b'fluffy tail'

┌──(kali㉿tamania)-[~]
└─$ 
```

Symmetric_server.py

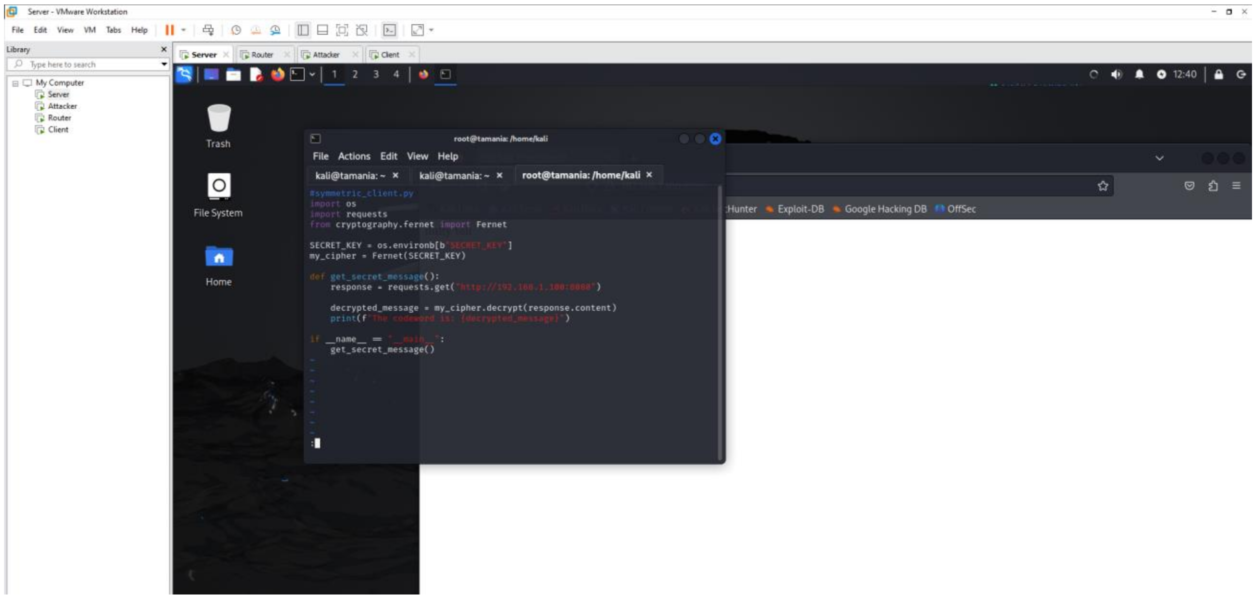


```
#symmetric_server.py
import os
from flask import Flask
from cryptography.fernet import Fernet

SECRET_KEY = os.environb[b"SECRET_KEY"]
SECRET_MESSAGE = b"fluffy tail"
app = Flask(__name__)

my_cipher = Fernet(SECRET_KEY)

@app.route("/")
def get_secret_message():
    return my_cipher.encrypt(SECRET_MESSAGE)
```

Symmetric_client.py



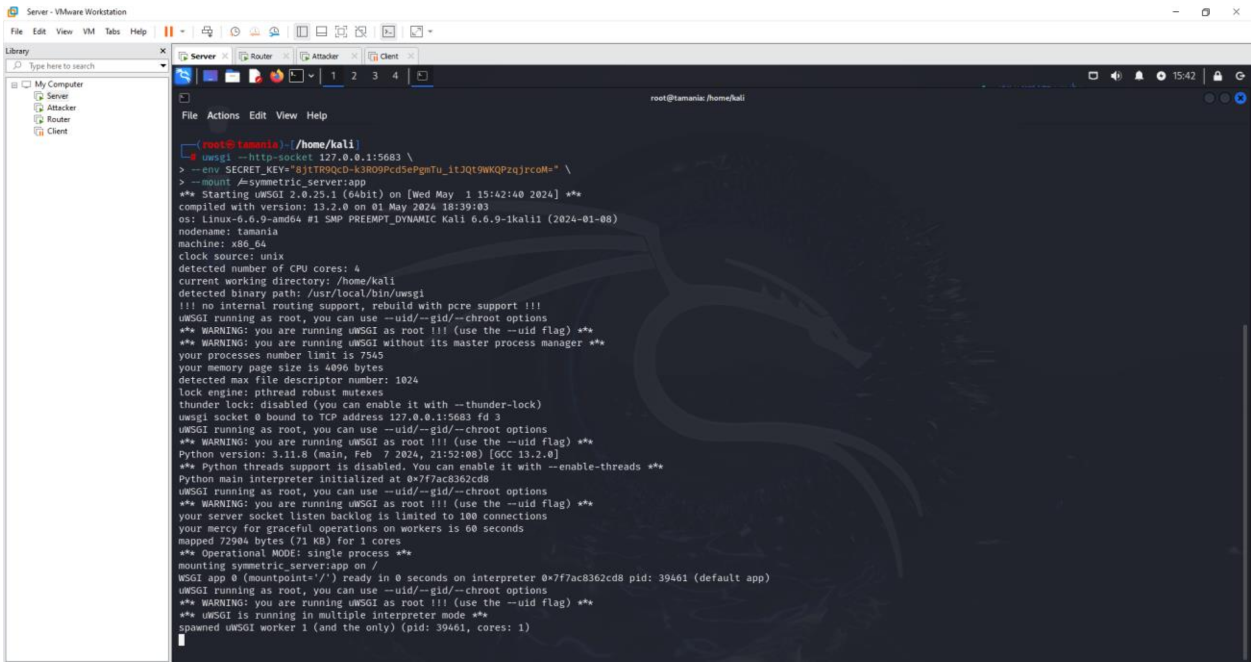
```
#symmetric_client.py
import os
import requests
from cryptography.fernet import Fernet

SECRET_KEY = os.environb[b"SECRET_KEY"]
my_cipher = Fernet(SECRET_KEY)

def get_secret_message():
    response = requests.get("http://127.0.0.1:5683")

    decrypted_message = my_cipher.decrypt(response.content)
    print(f"The codeword is: {decrypted_message}")

if __name__ == "__main__":
    get_secret_message()
```



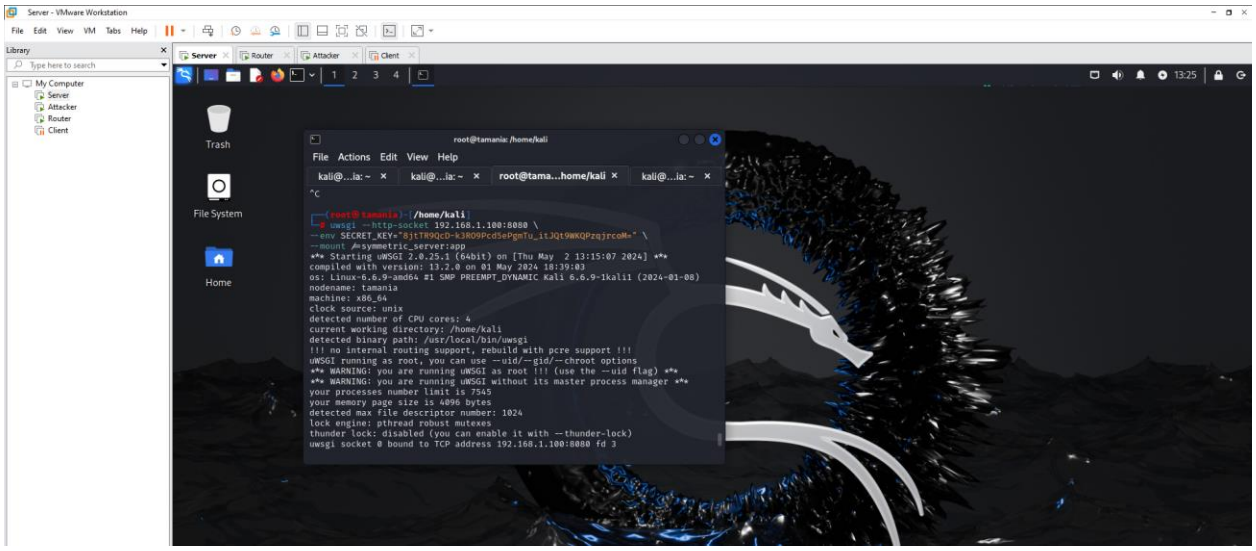
```
#symmetric_client.py
import os
import requests
from cryptography.fernet import Fernet

SECRET_KEY = os.environb[b"SECRET_KEY"]
my_cipher = Fernet(SECRET_KEY)

def get_secret_message():
    response = requests.get("http://192.168.1.100:8080")

    decrypted_message = my_cipher.decrypt(response.content)
    print(f"The codeword is: {decrypted_message}")

if __name__ == "__main__":
    get_secret_message()
:
```

NOTE: The python code is the exact same as provided in the website instructions except with some changes pertaining to my machine's IP address.

Run it

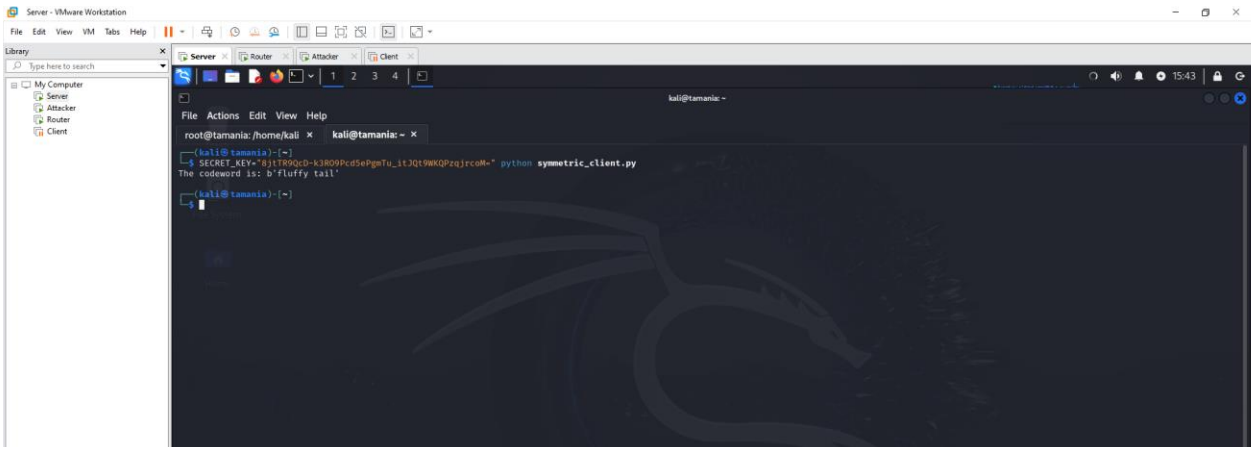


```
┌──(root㉿tamania)-[/home/kali]
└─# uwsgi --http-socket 127.0.0.1:5683 \
> --env SECRET_KEY="8jtTR9QcD-k3RO9Pcd5ePgmTu_itJQt9WKQPzqjrcoM=" \
> --mount /=symmetric_server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Wed May  1 15:42:40 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 127.0.0.1:5683 fd 3
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Python version: 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x7f7ac8362cd8
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 72904 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
mounting symmetric_server:app on /
WSGI app 0 (mountpoint='/') ready in 0 seconds on interpreter 0x7f7ac8362cd8 pid: 39461 (default app)
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 39461, cores: 1)
```



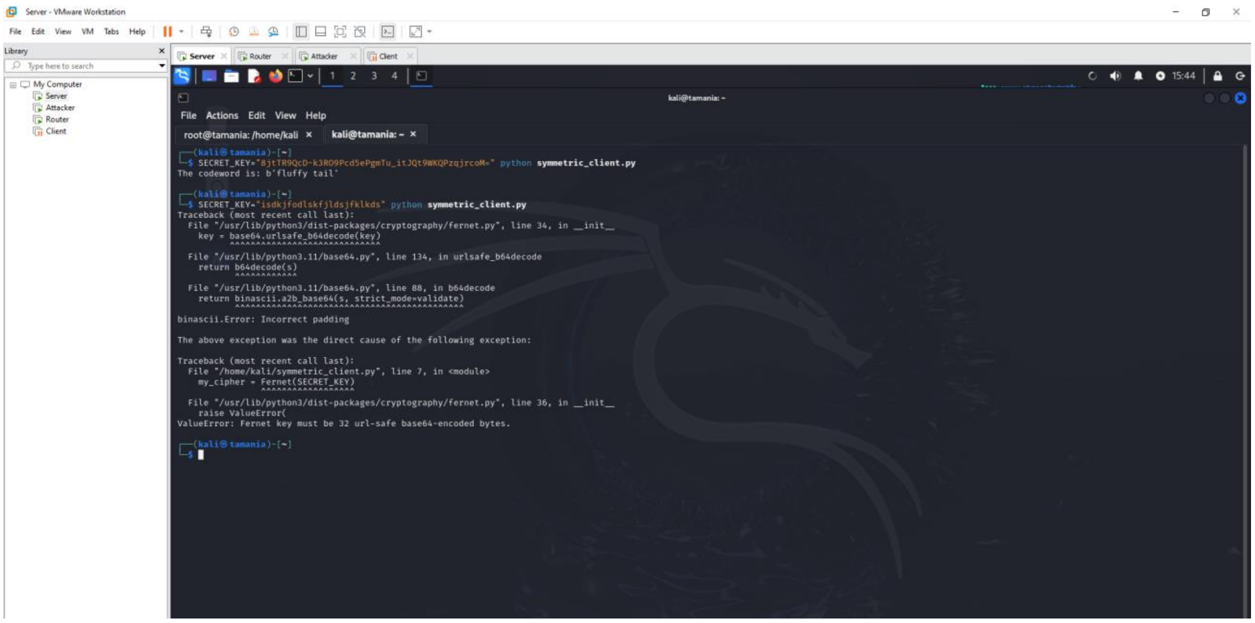
```
^C

┌──(root㉿tamania)-[/home/kali]
└─# uwsgi --http-socket 192.168.1.100:8080 \
--env SECRET_KEY="8jtTR9QcD-k3RO9Pcd5ePgmTu_itJQt9WKQPzqjrcoM=" \
--mount /=symmetric_server:app
*** Starting uWSGI 2.0.25.1 (64bit) on [Thu May  2 13:15:07 2024] ***
compiled with version: 13.2.0 on 01 May 2024 18:39:03
os: Linux-6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08)
nodename: tamania
machine: x86_64
clock source: unix
detected number of CPU cores: 4
current working directory: /home/kali
detected binary path: /usr/local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7545
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 192.168.1.100:8080 fd 3
```

Query it



```
┌──(kali㉿tamania)-[~]
└─$ SECRET_KEY="8jtTR9QcD-k3RO9Pcd5ePgmTu_itJQt9WKQPzqjrcoM=" python symmetric_client.py
The codeword is: b'fluffy tail'

┌──(kali㉿tamania)-[~]
└─$ 
```

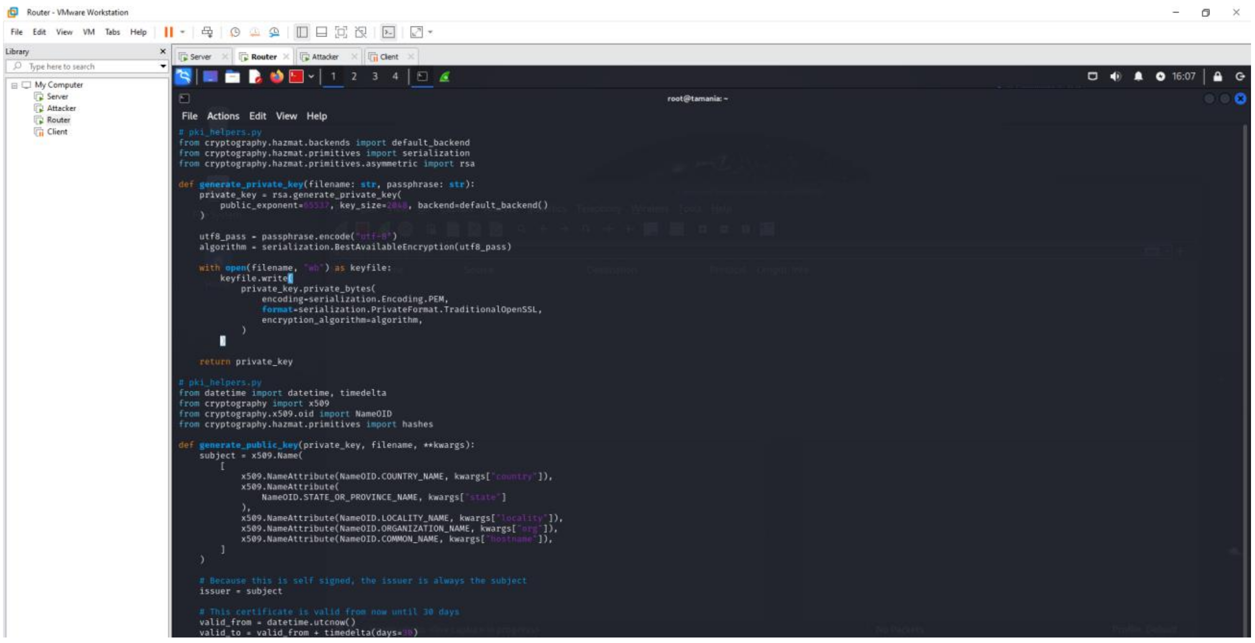
When using invalid secret key



```
┌──(kali㉿tamania)-[~]
└─$ SECRET_KEY="8jtTR9QcD-k3RO9Pcd5ePgmTu_itJQt9WKQPzqjrcoM=" python symmetric_client.py
The codeword is: b'fluffy tail'

┌──(kali㉿tamania)-[~]
└─$ SECRET_KEY="isdkjfodlskfjldsjfklkds" python symmetric_client.py
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/cryptography/fernet.py", line 34, in __init__
    key = base64.urlsafe_b64decode(key)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/usr/lib/python3.11/base64.py", line 134, in urlsafe_b64decode
    return b64decode(s)
           ^^^^^^^^^^^^

  File "/usr/lib/python3.11/base64.py", line 88, in b64decode
    return binascii.a2b_base64(s, strict_mode=validate)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
binascii.Error: Incorrect padding

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/home/kali/symmetric_client.py", line 7, in <module>
    my_cipher = Fernet(SECRET_KEY)
                ^^^^^^^^^^^^^^^^^^

  File "/usr/lib/python3/dist-packages/cryptography/fernet.py", line 36, in __init__
    raise ValueError(
ValueError: Fernet key must be 32 url-safe base64-encoded bytes.

┌──(kali㉿tamania)-[~]
└─$ 
```

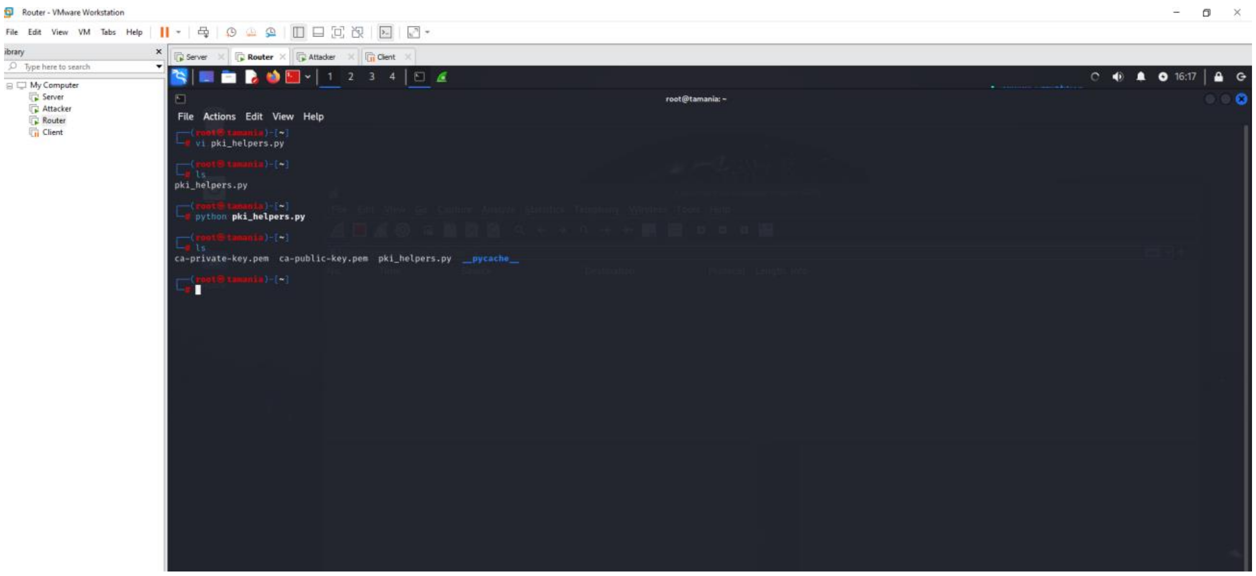
5. **(30 pts) Use certificate to verify public keys and enable safe data transfer**
    a. Use Router VM to simulate the Certificate Authority (CA) to issue self-signed public key
    b. Client and Server VMs generate Certificate Signing Requests (CSR) to CA
    c. CA issues certificates to Client and Server VMs
    d. Run HTTPs server and Client is able to make a connections to it

Using router VM to do CA



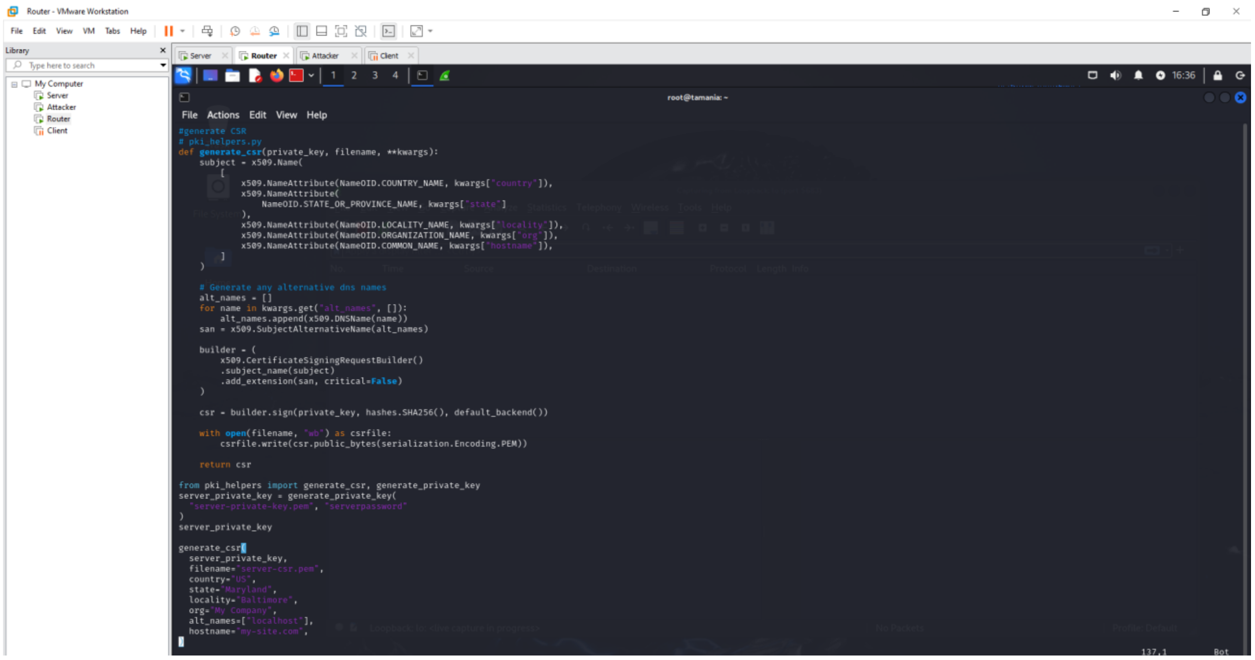
```
# pki_helpers.py
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa

def generate_private_key(filename: str, passphrase: str):
    private_key = rsa.generate_private_key(
        public_exponent=65537, key_size=2048, backend=default_backend()
    )

    utf8_pass = passphrase.encode("utf-8")
    algorithm = serialization.BestAvailableEncryption(utf8_pass)

    with open(filename, "wb") as keyfile:
        keyfile.write(
            private_key.private_bytes(
                encoding=serialization.Encoding.PEM,
                format=serialization.PrivateFormat.TraditionalOpenSSL,
                encryption_algorithm=algorithm,
            )
        )

    return private_key

# pki_helpers.py
from datetime import datetime, timedelta
from cryptography import x509
from cryptography.x509.oid import NameOID
from cryptography.hazmat.primitives import hashes

def generate_public_key(private_key, filename, **kwargs):
    subject = x509.Name(
        [
            x509.NameAttribute(NameOID.COUNTRY_NAME, kwargs["country"]),
            x509.NameAttribute(
                NameOID.STATE_OR_PROVINCE_NAME, kwargs["state"]
            ),
            x509.NameAttribute(NameOID.LOCALITY_NAME, kwargs["locality"]),
            x509.NameAttribute(NameOID.ORGANIZATION_NAME, kwargs["org"]),
            x509.NameAttribute(NameOID.COMMON_NAME, kwargs["hostname"]),
        ]
    )

    # Because this is self signed, the issuer is always the subject
    issuer = subject

    # This certificate is valid from now until 30 days
    valid_from = datetime.utcnow()
    valid_to = valid_from + timedelta(days=30)
```

After importing helper functions, generate and save to pem files. Shown here in directory.



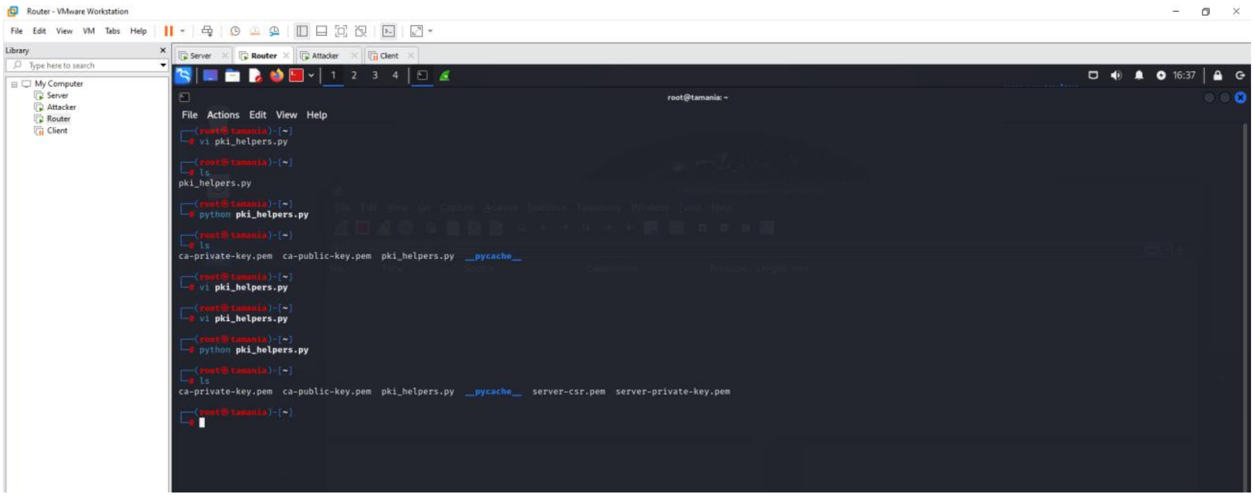
```
┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__

┌──(root㉿tamania)-[~]
└─# 
```

Generate CSR into pki_helpers.py



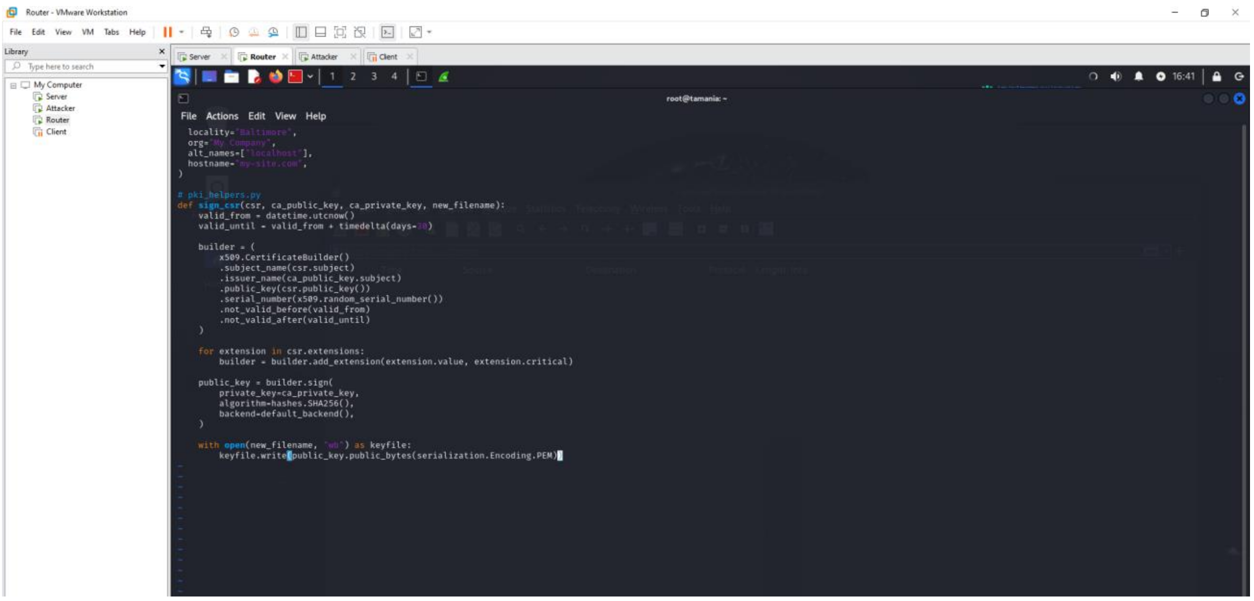
```
#generate CSR
# pki_helpers.py
def generate_csr(private_key, filename, **kwargs):
    subject = x509.Name(
        [
            x509.NameAttribute(NameOID.COUNTRY_NAME, kwargs["country"]),
            x509.NameAttribute(
                NameOID.STATE_OR_PROVINCE_NAME, kwargs["state"]
            ),
            x509.NameAttribute(NameOID.LOCALITY_NAME, kwargs["locality"]),
            x509.NameAttribute(NameOID.ORGANIZATION_NAME, kwargs["org"]),
            x509.NameAttribute(NameOID.COMMON_NAME, kwargs["hostname"]),
        ]
    )

    # Generate any alternative dns names
    alt_names = []
    for name in kwargs.get("alt_names", []):
        alt_names.append(x509.DNSName(name))
    san = x509.SubjectAlternativeName(alt_names)

    builder = (
        x509.CertificateSigningRequestBuilder()
        .subject_name(subject)
        .add_extension(san, critical=False)
    )

    csr = builder.sign(private_key, hashes.SHA256(), default_backend())

    with open(filename, "wb") as csrfile:
        csrfile.write(csr.public_bytes(serialization.Encoding.PEM))

    return csr

from pki_helpers import generate_csr, generate_private_key
server_private_key = generate_private_key(
  "server-private-key.pem", "serverpassword"
)
server_private_key

generate_csr(
  server_private_key,
  filename="server-csr.pem",
  country="US",
  state="Maryland",
  locality="Baltimore",
  org="My Company",
  alt_names=["localhost"],
  hostname="my-site.com",
)
```

Two files server-private-key.pem and server-csr.pem



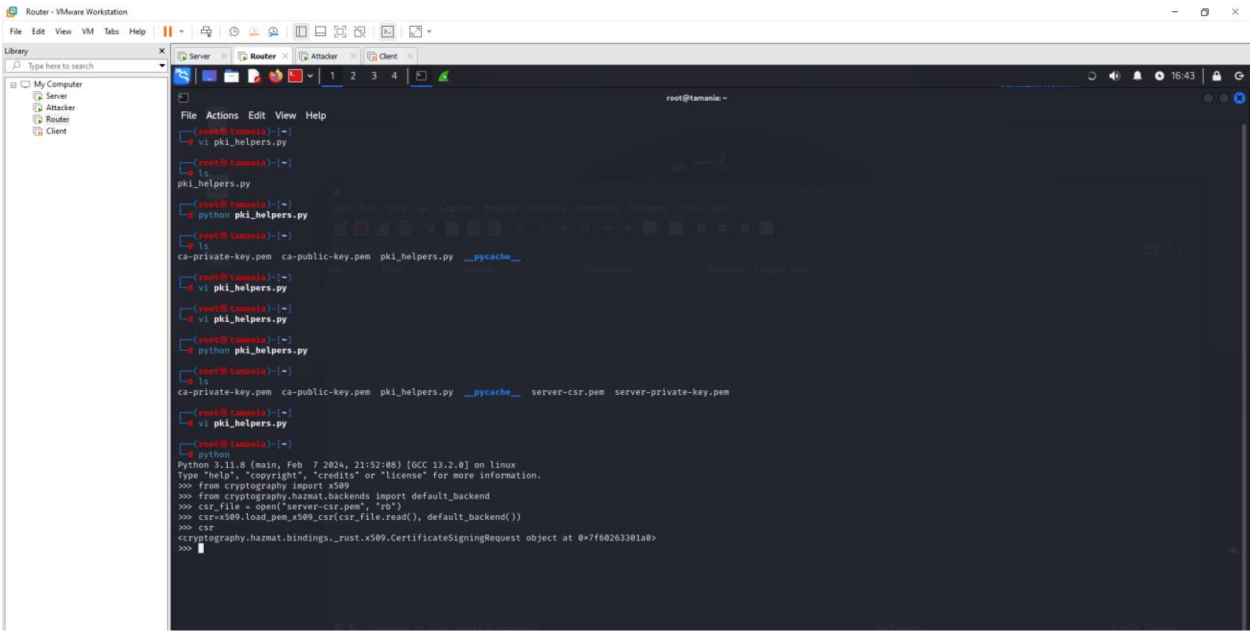
```
┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__  server-csr.pem  server-private-key.pem

┌──(root㉿tamania)-[~]
└─# 
```

Create own pub key



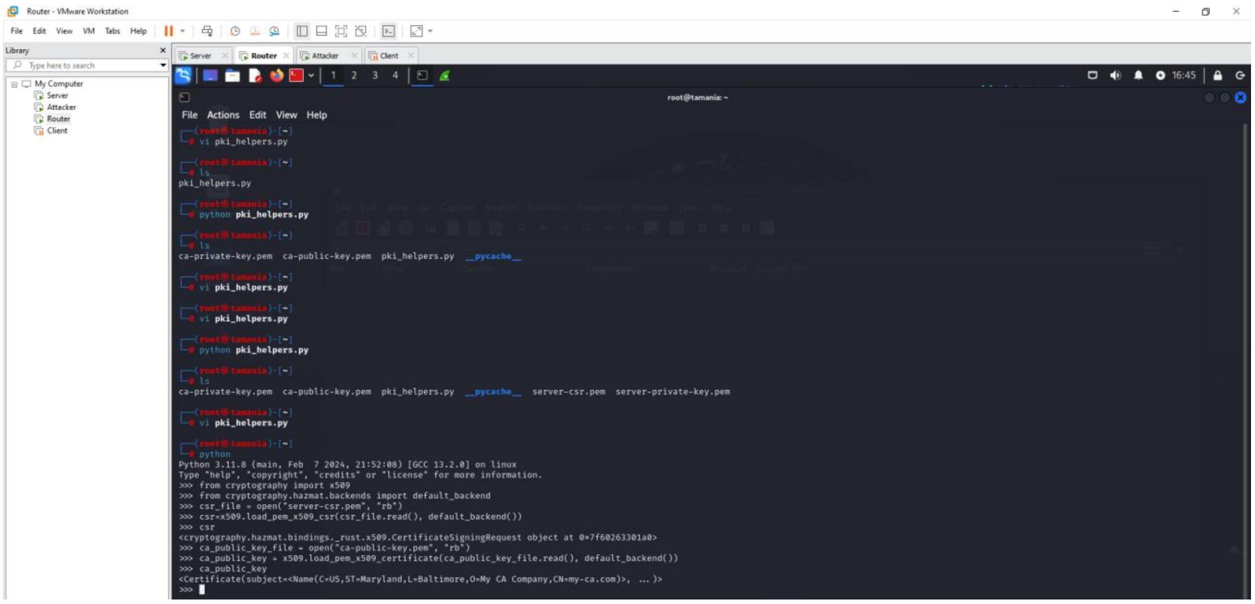
```
    locality="Baltimore",
    org="My Company",
    alt_names=["localhost"],
    hostname="my-site.com",
)

# pki_helpers.py
def sign_csr(csr, ca_public_key, ca_private_key, new_filename):
    valid_from = datetime.utcnow()
    valid_until = valid_from + timedelta(days=30)

    builder = (
        x509.CertificateBuilder()
        .subject_name(csr.subject)
        .issuer_name(ca_public_key.subject)
        .public_key(csr.public_key())
        .serial_number(x509.random_serial_number())
        .not_valid_before(valid_from)
        .not_valid_after(valid_until)
    )

    for extension in csr.extensions:
        builder = builder.add_extension(extension.value, extension.critical)

    public_key = builder.sign(
        private_key=ca_private_key,
        algorithm=hashes.SHA256(),
        backend=default_backend(),
    )

    with open(new_filename, "wb") as keyfile:
        keyfile.write(public_key.public_bytes(serialization.Encoding.PEM))
```

Load csr



```
┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__  server-csr.pem  server-private-key.pem

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cryptography import x509
>>> from cryptography.hazmat.backends import default_backend
>>> csr_file = open("server-csr.pem", "rb")
>>> csr=x509.load_pem_x509_csr(csr_file.read(), default_backend())
>>> csr
<cryptography.hazmat.bindings._rust.x509.CertificateSigningRequest object at 0x7f60263301a0>
>>> 
```

Load CA’s public key



```
┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python pki_helpers.py

┌──(root㉿tamania)-[~]
└─# ls
ca-private-key.pem  ca-public-key.pem  pki_helpers.py  __pycache__  server-csr.pem  server-private-key.pem

┌──(root㉿tamania)-[~]
└─# vi pki_helpers.py

┌──(root㉿tamania)-[~]
└─# python
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cryptography import x509
>>> from cryptography.hazmat.backends import default_backend
>>> csr_file = open("server-csr.pem", "rb")
>>> csr=x509.load_pem_x509_csr(csr_file.read(), default_backend())
>>> csr
<cryptography.hazmat.bindings._rust.x509.CertificateSigningRequest object at 0x7f60263301a0>
>>> ca_public_key_file = open("ca-public-key.pem", "rb")
>>> ca_public_key = x509.load_pem_x509_certificate(ca_public_key_file.read(), default_backend())
>>> ca_public_key
<Certificate(subject=<Name(C=US,ST=Maryland,L=Baltimore,O=My CA Company,CN=my-ca.com)>, ...)>
>>> 
```

Load private key

I didn’t finish this portion of the project.