**NCTU CN2018 Lab. 1 – Packet Manipulation via Scapy**

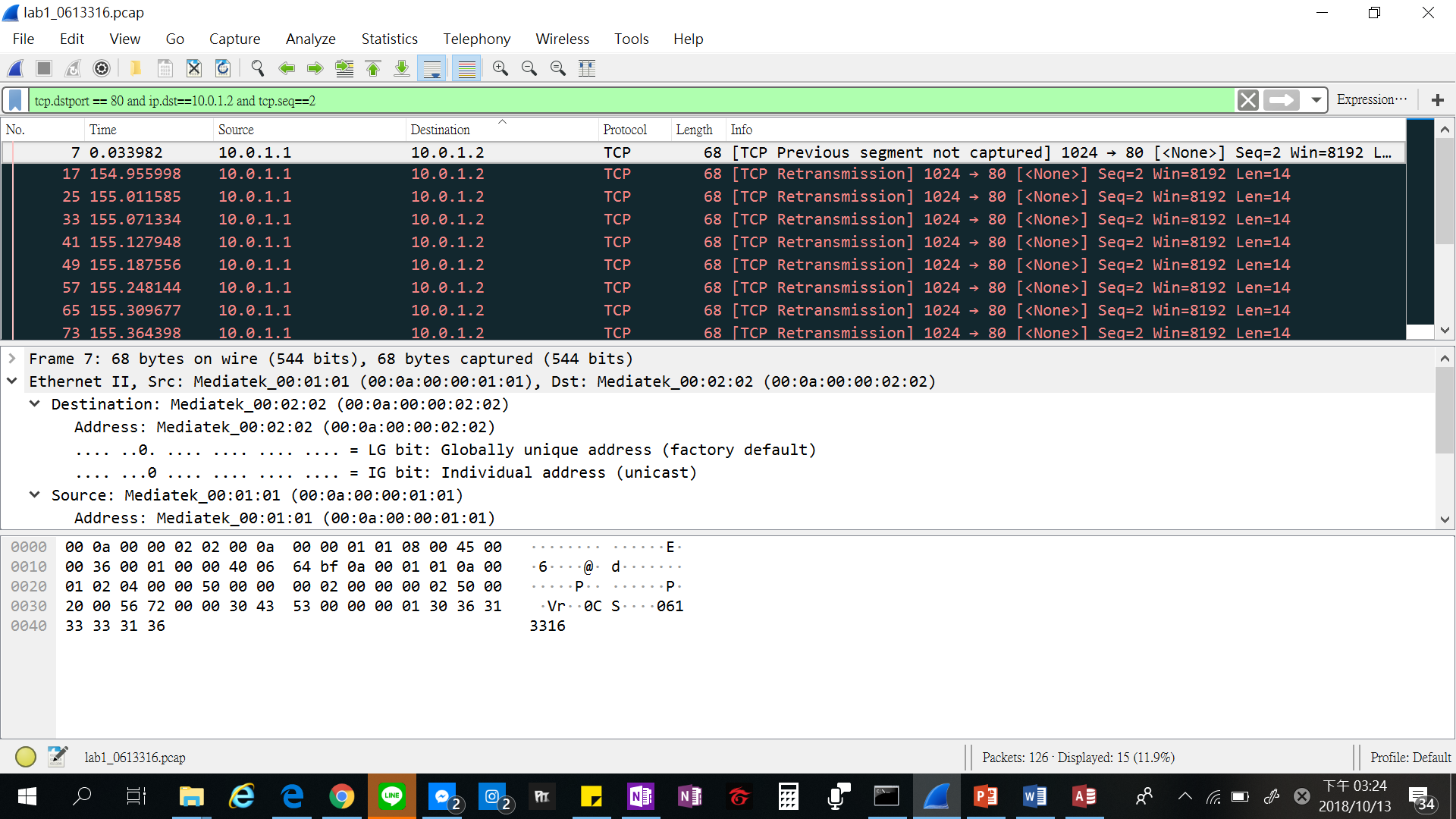
**Student name:**曾筱晴  **Student ID:** 0613316 **Department**: CS

**Part A. Questions**

1. What is your command to filter the packet with customized header on Wireshark?

tcp.dstport == 80 and ip.dst==10.0.1.2 and tcp.seq==2

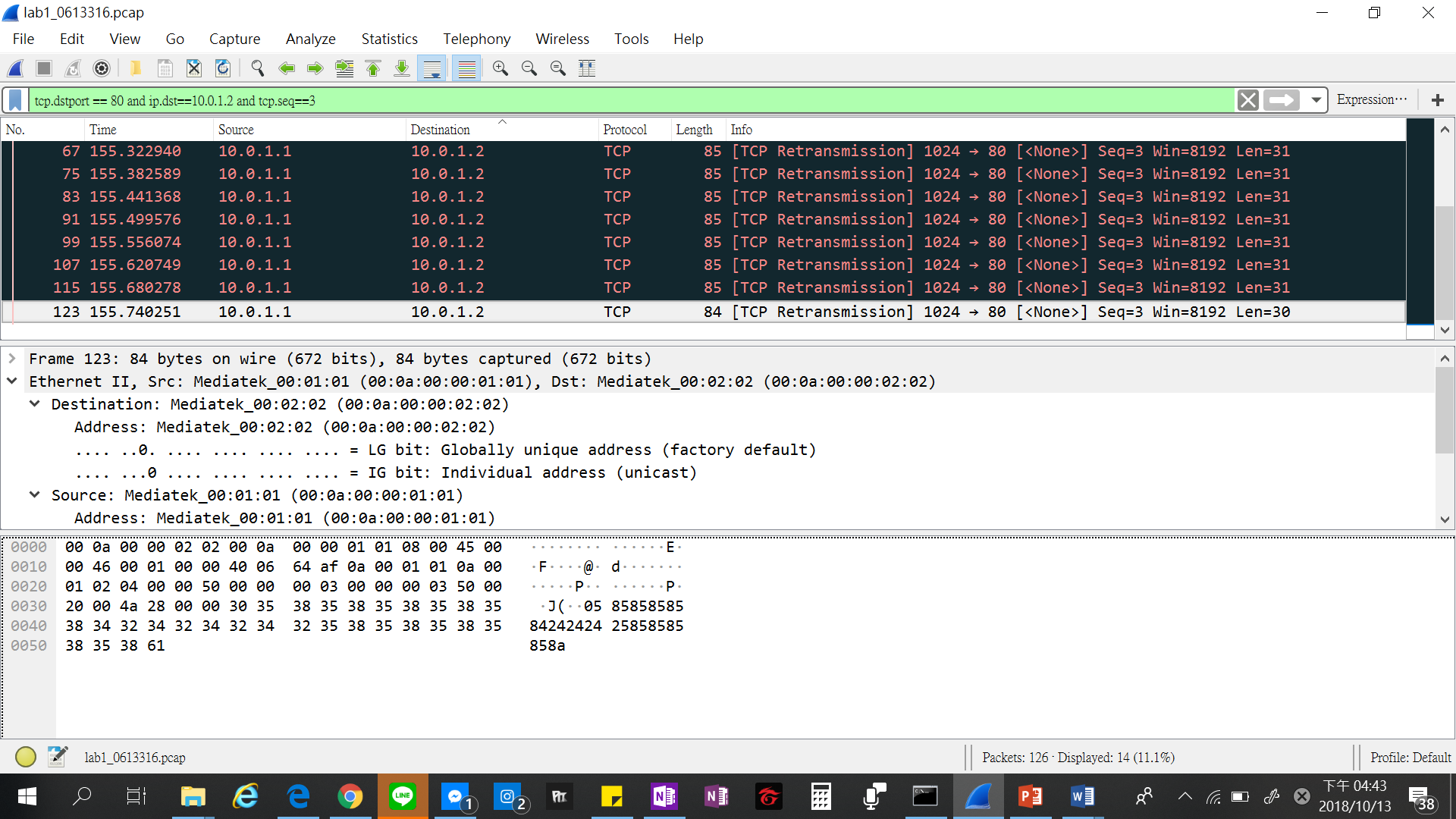
2. Show the screenshot of filtering the packet with customized header.



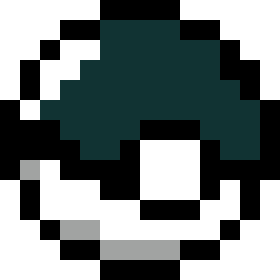
3. What is your command to filter the packet with “secret” payload on Wireshark?

tcp.dstport == 80 and ip.dst==10.0.1.2 and tcp.seq==3

4. Show the screenshot of filtering the packet with “secret” payload.



5. Show the result after decoding the “secret” payload.



**Part B. Description**

**Task 1 – Environment setup**

• Download files and setup Github

1. Download required files from GitHub

$ git clone https://github.com/yungshenglu/Packet\_Manipulation

1. Get and set repository or global options

$ git config --global user.name "isbecky27"

$ git config --global user.email "is.becky27@gmail.com"

1. Set a new remote URL to your repository

$ git remote set-url origin https://github.com/nctucn/lab1-isbecky27.git

1. Push your repository to remote

$ git push origin master

• Configure Dockerfile

1. Download base image from yungshenglu/ubuntu-env:16.04

FROM yungshenglu/ubuntu-env:16.04

1. Update all software repository

FROM apt-get update

1. Install software repository

RUN apt-get install –y tcpdump

1. Install pip packages

RUN pip install scapy

1. Set the container listens on the specified ports at runtime

EXPOSE 22

1. Clone the repository from GitHub

RUN git clone https://github.com/yungshenglu/Packet\_Manipulation.git

• Build the container with Dockerfile

1. Build the image from Dockerfile

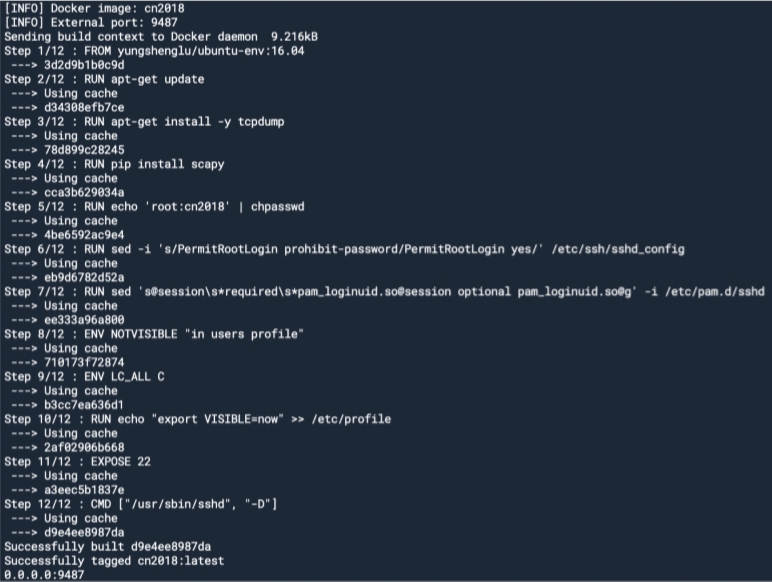
docker build -t cn2018 .

1. Build a container named cn2018\_c from cn2018

docker run -d -p 9487:22 --privileged --name cn2018\_c cn2018

1. List port 22 mapping on cn2018\_c

docker port cn2018\_c 22

1. Get the following result if succeed

• Login to Docker container

1. Open PieTTY and connect to the Docker

IP address: 127.0.0.1

Port: 9487

1. Login as root

Login : root

Password : cn2018

• Create the namespace in ./src/scripts/main.sh for h2

1. Create h2 network namespaces

ip netns add h2

1. Delete h2 network namespaces

ip netns del h2

1. Bring up the lookup interface in h2

ip netns exec h2 ip link set lo up

1. Set the interface of h2 to h2-eth0

ip link set h2-eth0 netns h2

1. Delete the interface of h2-eth0

ip link delete h2-eth0

1. Activate h2-eth0 and assign IP address

ip netns exec h2 ip link set dev h2-eth0 up

ip netns exec h2 ip link set h2-eth0 address 00:0a:00:00:02:02

ip netns exec h2 ip addr add 10.0.1.2/24 dev h2-eth0

1. Disable all IPv6 on h2-eth0

ip netns exec h2 sysctl net.ipv6.conf.h2-eth0.disable\_ipv6=1

1. Set the gateway of h2 to 10.0.1.254

ip netns exec h2 ip route add default via 10.0.1.254

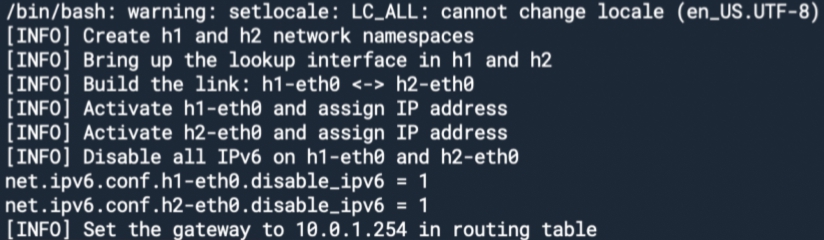
• Run main.sh to build the namespace

1. Execute the following commands

$ chmod +x main.sh

$ ./main.sh net

1. Get the following result if succeed

****

**Task 2. Define protocol via Scapy**

• Define my protocol

1. Open ./src/Protocal.py and add the following code
2. Set the name of protocol

name = 'Student'

1. Define the fields in protocol

fields\_desc = [

StrField('index', '0'),

StrField('dept', 'cs', fmt = 'H', remain = 0),

IntEnumField('gender', 2, {

1: 'female',

2: 'male'

}),

StrField('id', '000000', fmt = 'H', remain = 0),

]

**Task 3 – Send packets**

• Set my own packet header in ./src/sender.py

1. Open ./src/sender.py and add the following code
2. Set source and destination IP address

src\_ip = '10.0.1.1'

dst\_ip = '10.0.1.2'

1. Set source and destination port

src\_port = 1024

dst\_port = 80

1. Define IP header

ip = IP(src = src\_ip, dst = dst\_ip)

1. Define customized header

my\_id = '0613316'

my\_dept = 'CS'

my\_gender = 1

student = Protocol(id = my\_id, dept = my\_dept, gender = my\_gender)

• Send packets

1. Add the following code in ./src/sender.py
2. TCP connection – ACK

ack = tcp\_syn\_ack.seq + 1

tcp\_ack = TCP(sport = src\_port, dport = dst\_port, flags = 'A', seq = 1, ack = ack)

packet = ip / tcp\_ack

send(packet)

print '[INFO] Send ACK'

1. Send packet with customized header

ack = tcp\_ack.seq + 1

tcp = TCP(sport = src\_port, dport = dst\_port, flags = '', seq = 2, ack = ack)

packet = ip / tcp / student

send(packet)

print '[INFO] Send packet with customized header'

1. Send packet with secret payload

ack = tcp.seq + 1

tcp = TCP(sport = src\_port, dport = dst\_port, flags = '', seq = 3, ack = ack)

payload = Raw(secret[i]) packet = ip / tcp / payload

send(packet)

print '[INFO] Send packet with secret payload'

**Task 4 – Sniff packets**

• Receive and sniff packets

1. Open ./src/receiver.py and add the following code
2. Set source IP address and destination interface

dst\_iface = 'h2-eth0'

src\_ip = '10.0.1.1'

1. Sniff packets on destination interface

print '[INFO] Sniff on %s' % dst\_iface

packets = sniff(iface = dst\_iface, prn = lambda x: packetHandler(x))

1. Dump the sniffed packet into PCAP file

print '[INFO] Write into PCAP file'

filename = './out/lab1\_0' + id + '.pcap'

wrpcap(filename, packets)

**Task 5 – Run sender and receiver**

• Open tmux with horizontal two panes

1. Keep the path in ./src/
2. Open tmux

$tmux

1. Open new pane in horizontal

Ctrl-b Shift-%

• Switch into two namespaces

1. Run namespace h1 in your left pane

Ctrl-b Arrow-left key

$ ./scripts/main.sh run h1

1. Run namespace h2 in your right pane

Ctrl-b Arrow-right key

$ ./scripts/main.sh run h2

• Run receiver.py first

1. Switch between two panes

Ctrl-b Arrow-right key

1. Run receiver.py

h2> python receiver.py

• Run sender.py

1. Switch between two panes

Ctrl-b Arrow-left key

1. Run sender.py

h1> python sender.py

• Use tcpdump to show your PCAP file

1. Dump the PCAP via tcpdump

$ tcpdump -qns 0 -X -r lab1\_0613316.pcap

1. Then get a lab1\_0613316.pcap and recv\_secret.txt in ./src/out/

**Task 6 – Push your files to remote**

• Push your image to Docker Hub

1. Create a new image from a container’s changes

$ docker commit cn2018\_c isbecky27/cn2018\_lab1

1. Login to your Docker registry

$ docker login

1. Push an image to a registry

$ docker push isbecky27/cn2018\_lab1

• Push your files to GitHub

1. Get and set repository or global options

$ git config --global user.name "<NAME>"

$ git config --global user.email "<EMAIL>"

1. Add my files into staging area

$ git add .

1. Commit the files

$ git commit –m "Commit lab1 in class”

1. Set the remote URL to my remote repository

$ git remote set-url origin

<https://github.com/nctucn/lab1-0613316.git>

1. Push the files to remote repository

$ git push origin master

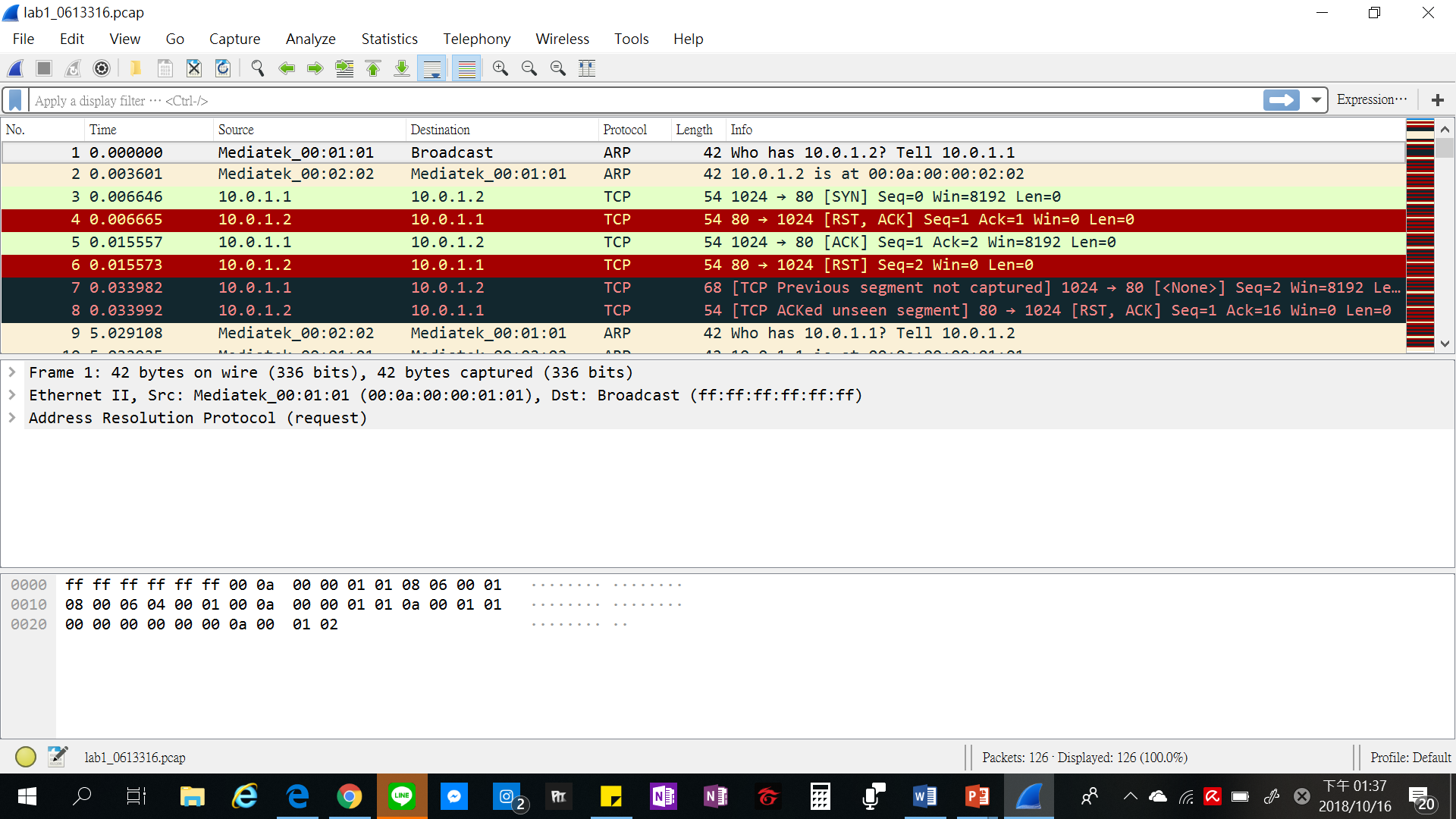
**Task 7 – Load PCAP via Wireshark**

• Download the code from GitHub

$ git clone <https://github.com/nctucn/lab1-yungshenglu.git>

• Install Wireshark 2.6.3

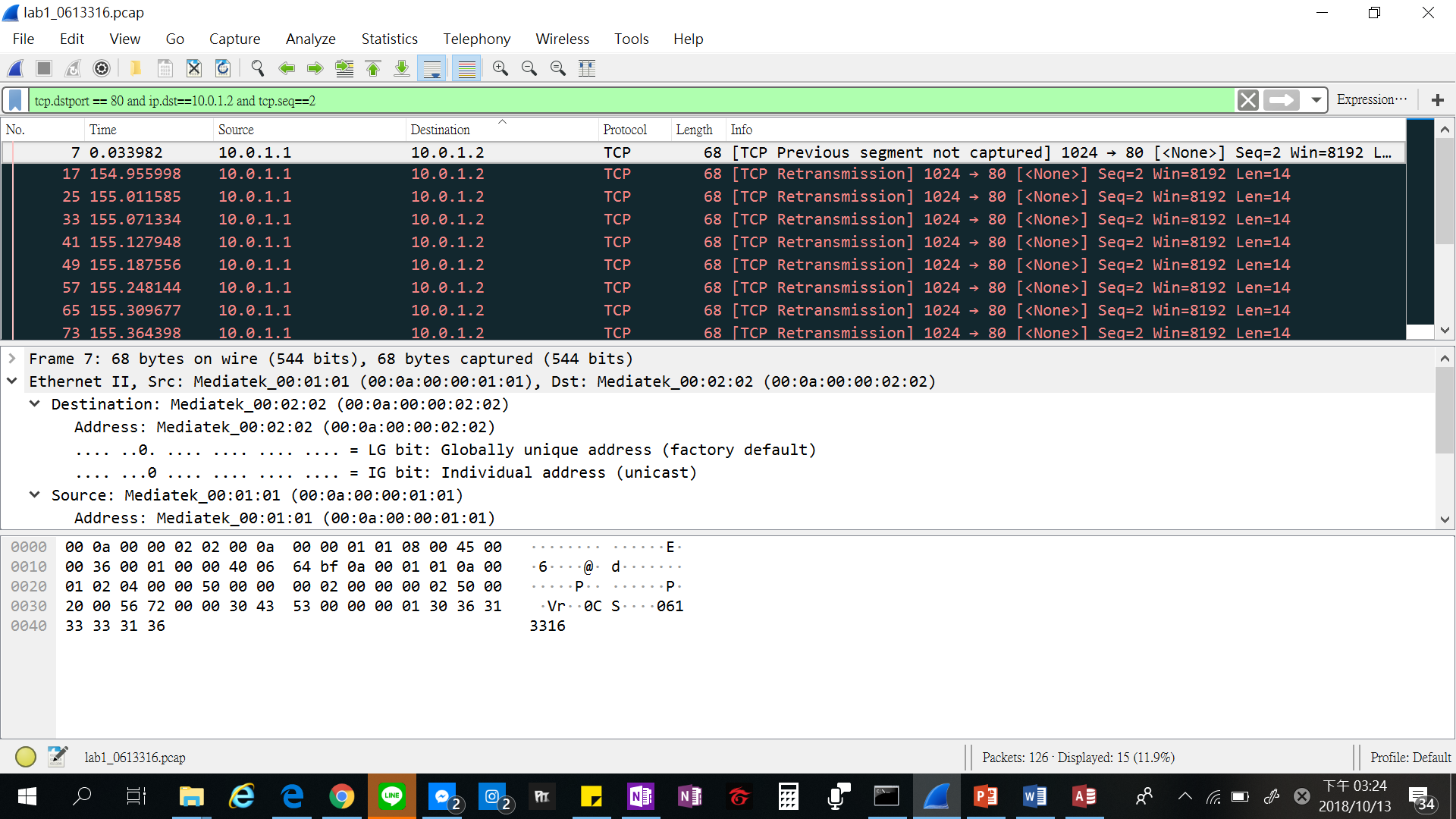
• Open the PCAP file using Wireshark



**Task 8 – Filter the target packet**

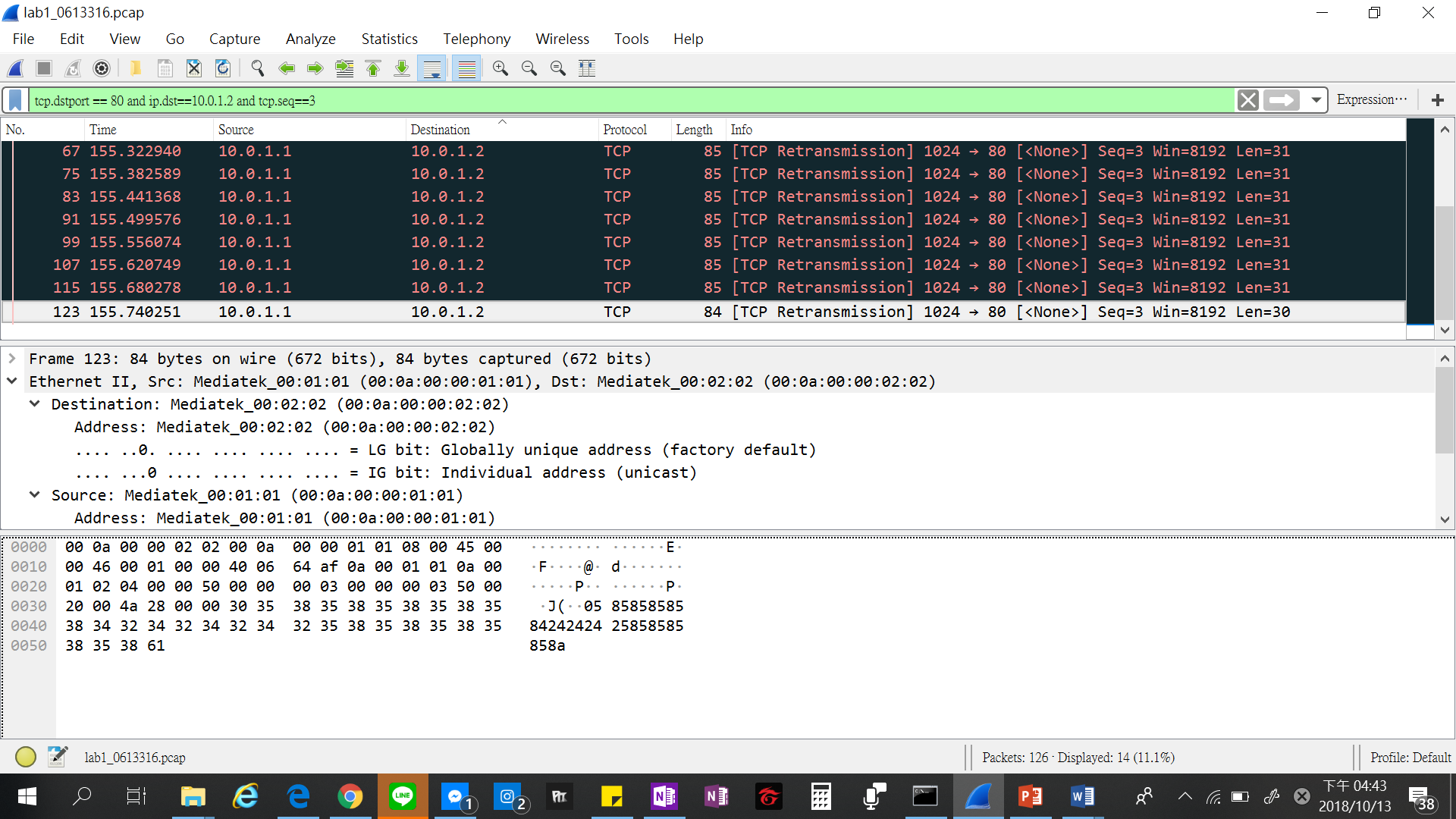
• Filter the packets of our defined protocol

Filter rule: tcp.dstport == 80 and ip.dst==10.0.1.2 and tcp.seq==2



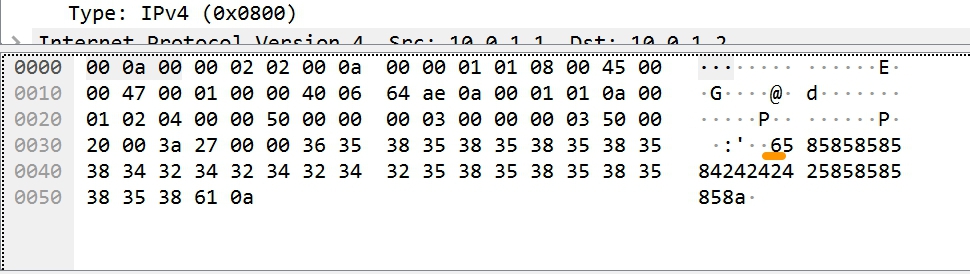
• Filter the packets with the “secret” bits

Filter rule: tcp.dstport == 80 and ip.dst==10.0.1.2 and tcp.seq==3



• What is my secret key? How to find it?

1. Filter packets first and get 14 secret packets
2. Combine the first digit in a secret payload (following screenshot) into 14 digits from 14 secret packets
3. Find my secret key is 61331606133160



**Task 9 – Decode the secret key**

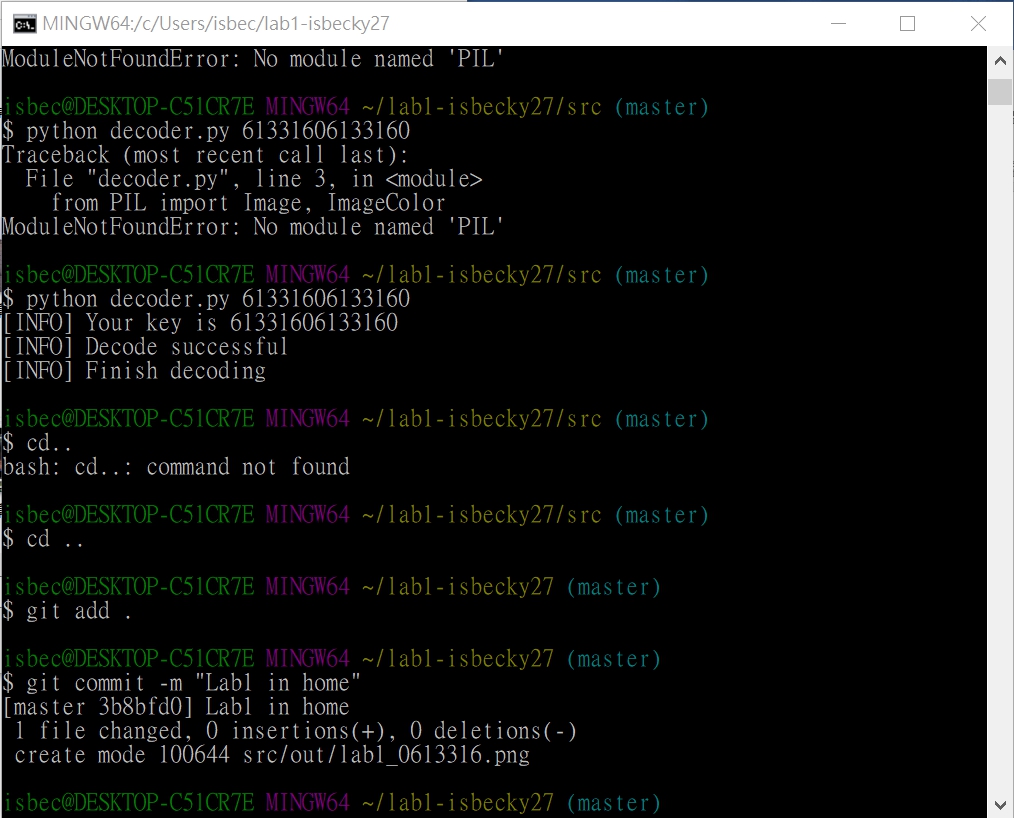
• Input the secret key into ./src/decoder.py on local machine

1. install PIL $ pip install Pillow
2. execute $ python decoder.py 6331606133160
3. get the following result if succeed

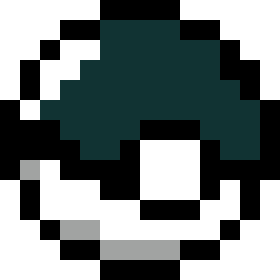
[INFO] Your key is 61331606133160

[INFO] Decode successful

[INFO] Finish decoding



1. get an image related to Pokemon in the file ./src/out/



**Part C. Bonus**

• What you have learned in this lab?

實作這次lab讓我學到了很多，從最一開始Docker的環境設定，定義自己的協定、設定自己的封包標頭，在PieTTY上修改程式碼，使用tmux讓終端機一次使用多個程式，一端傳送封包和一端接收封包等，以及該如何使用Wireshark來過濾封包，之後再藉由過濾封包得出的密碼來解碼圖片，最後將所有結果上傳到Github。雖然做完需要花費蠻多時間的，但過程蠻有趣的，可以學到很多東西。

• What difficulty you have met in this lab?

在實際操作的時候，會發現其實有很多小細節需要注意。

像是Task2~4的程式碼一旦有錯，Task5在傳送接收封包的時候就會有問題產生。

Task6要將檔案傳到Github上時，會發現pcap檔上傳不上去。

Task8在過濾封包時指令需上網查清楚，然後用對的方式去過濾封包，不然會產生錯誤或者過濾不出正確的封包。

Task9則是需要記得PIL，不然在解碼時會出現，電腦無PIL的模組無法產生圖片的錯誤訊息。