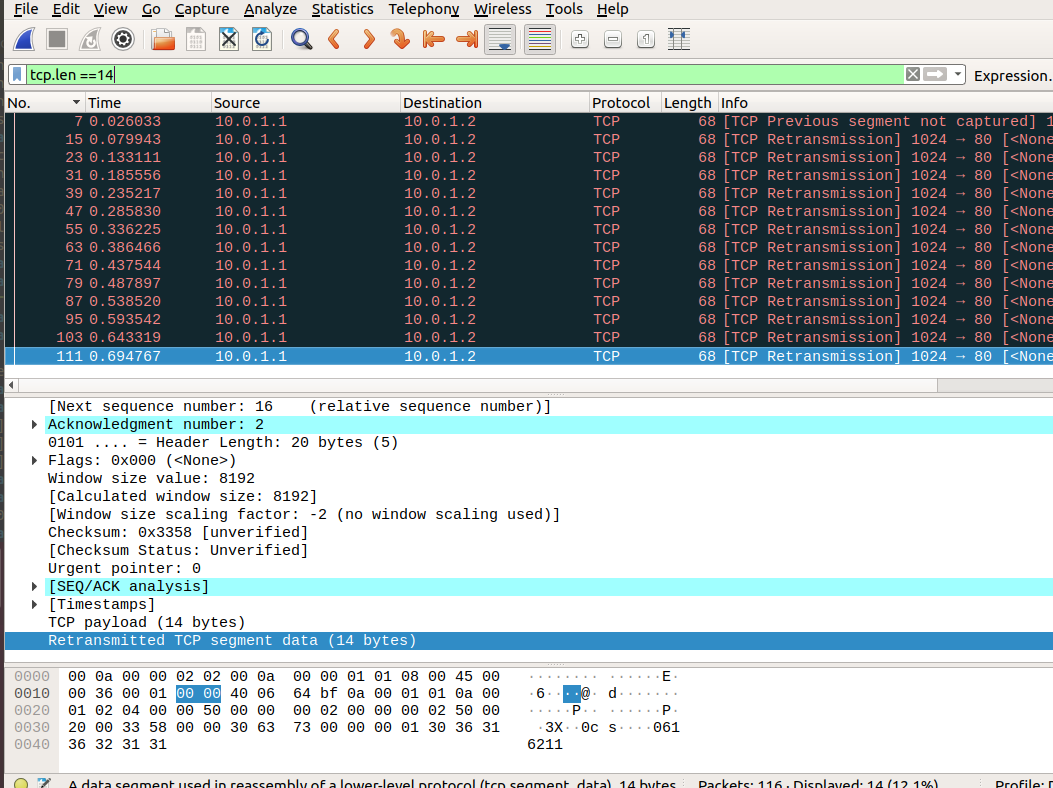
**NCTU CN 2018 lab. 1 – Packet Manipulation via Scapy**

Student name: 吳宜珈 ID: 0616211

**Question**

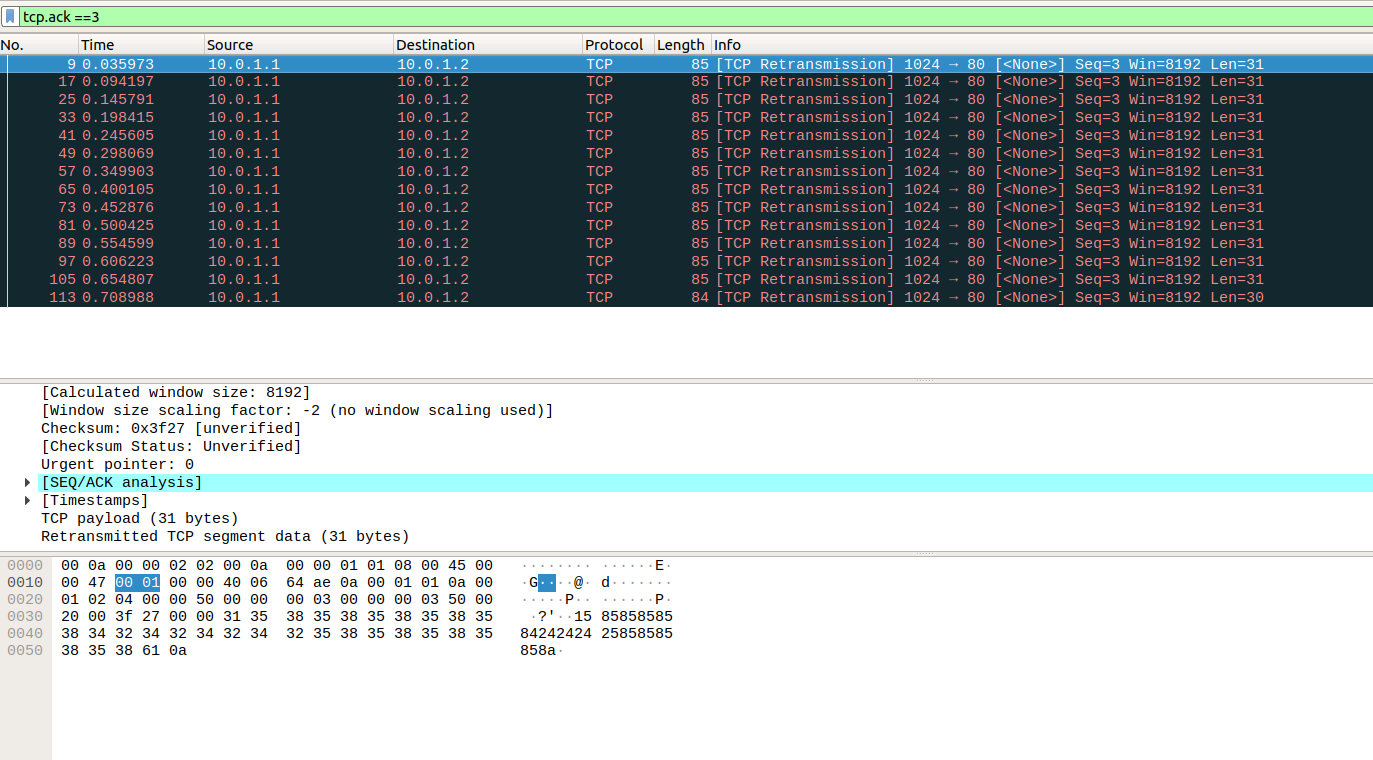
1. command: **tcp.len == 14**

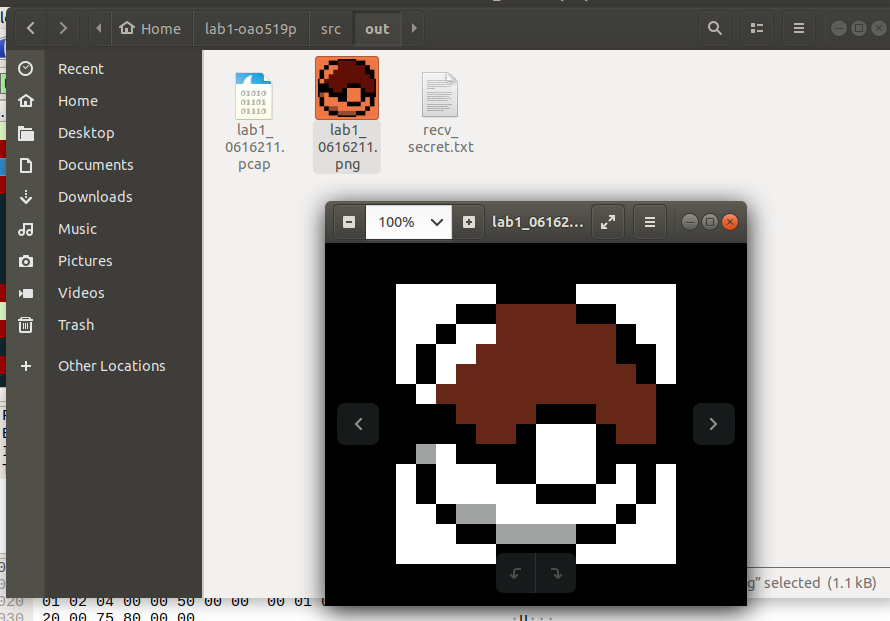
2. get 14 data with my ID at last 7 digits number.



3. command: **tcp.ack ==3**

4. Use filter command to get datas with secret key



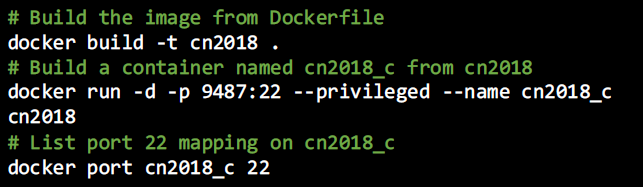
5. result after decode: 

**Step Detail**

**Task 1 – Enviroment setup**

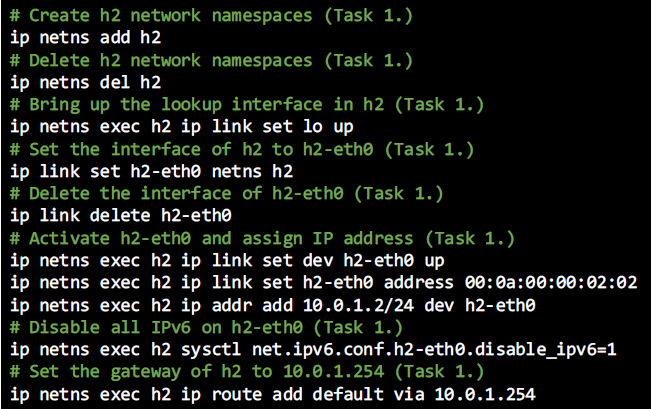
Use lots of commands to set up (github & docker&scapy)

And build image from Dockerfile & container from cn2018



Login docker by ssh

Enter main.sh and create the namespace of h2 part(h1 was done by TA)

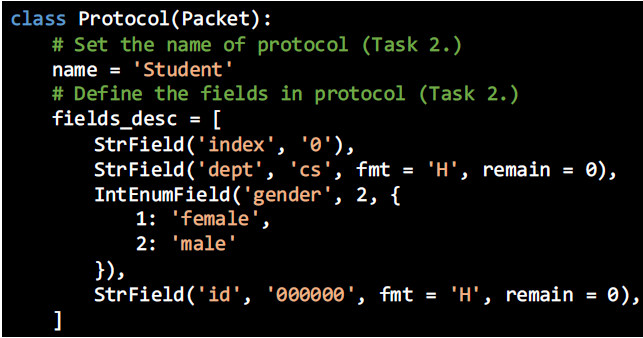


Run main.sh and build the namespace successfully.

**Task 2 – Define protocol via Scapy**

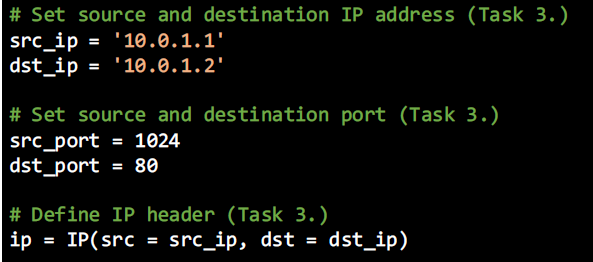
Scapy 是由 Python 寫出的一款強大、著名的網路封包分析函式，但它不只是函式，同時也是一個基於 Python交互式(interactive)的工具程式。目前比較有名跟它類似的工具是Wireshark，不過因為它可以**配合Python** 寫出俱有特殊目的的封包分析方式，也允許你客製自己的用法甚至自訂 protocol。 (copy from google)

That’s why we use **Scapy** to **define protocol** in this task.

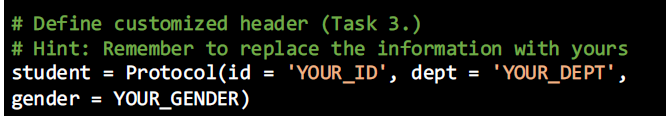


**Task 3 – send packets**

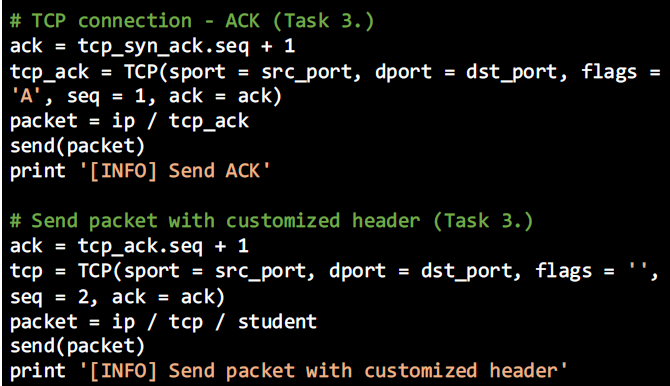
Set source and destination ‘s information. (will be seen different objects later)



Set my own packet header ( same format as we set in task 2)

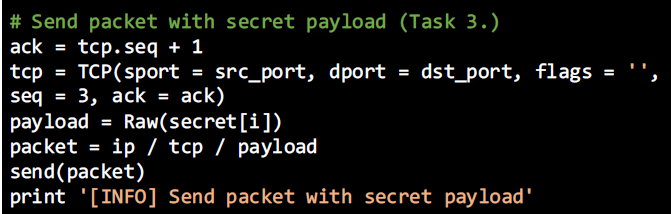


Set up the TCP connection.



From right picture we can know what is ACK number(and other TCP headers )

The term '**payload**' is used to distinguish between the 'interesting' information in a chunk of data or similar, and the overhead to support it. In programming, the most common usage of the term is in the context of message protocols, to differentiate the protocol overhead from the actual data.

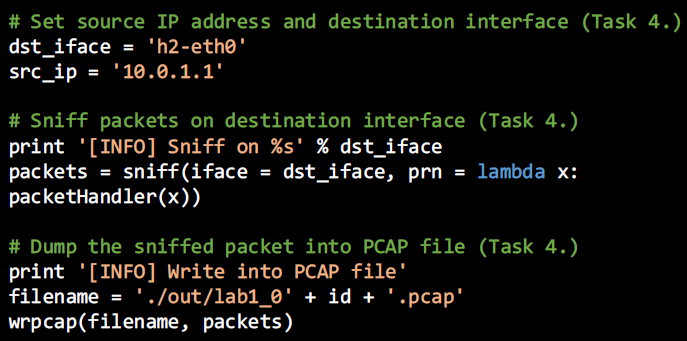


So by this codes, we will later send our secret payload and get secret key.

**Task 4 – Sniff packets**

**Packet sniffe**r is a computer program or piece of computer hardware that can intercept and log traffic that passes over a digital network or part of a network. Packet capture is the process of intercepting and logging traffic. As data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications.

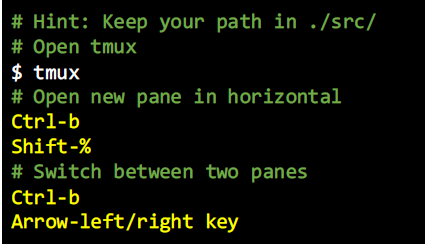
After sniffing the packet, dump it into PCAP file. (for later use)



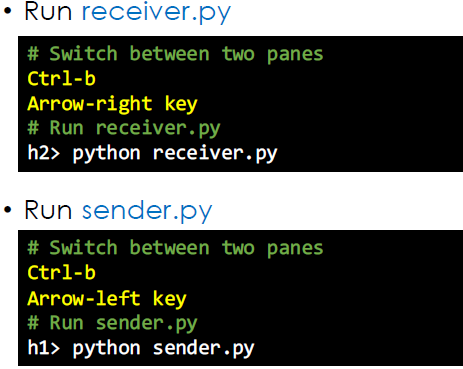
**Task 5 – Run sender and receiver**

Use tmux command to split two horizontal panes,

and use Ctrl-b + right/left key to switch.



Left: h1(receiver) Right:h2(sender) each side run each .py then get .pcap & .txt

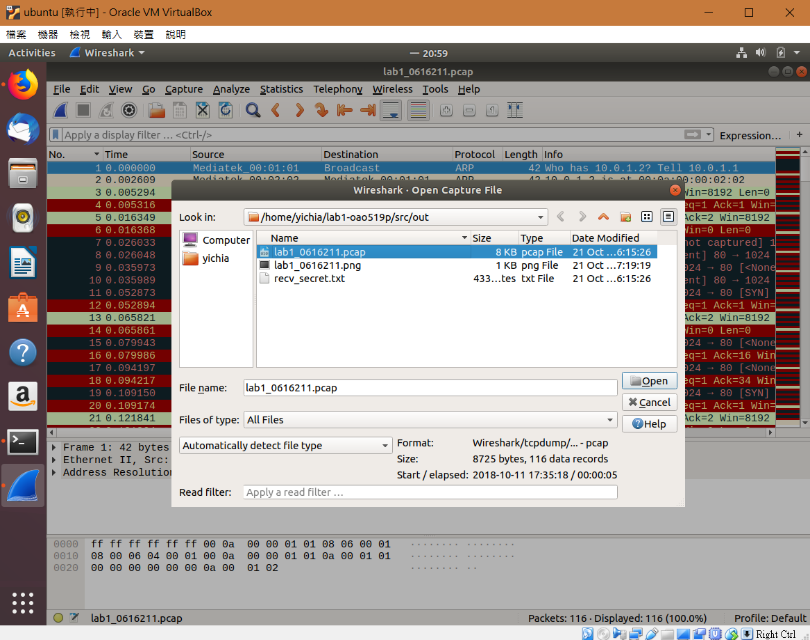


**Task 6 – Push your files to remote**

Push image to Docker Hub & files to Github. (suppose no need to explain)

**Task 7 – Load PCAP via Wireshark**

Downlond the code from github, then open the PCAP with Wireshark

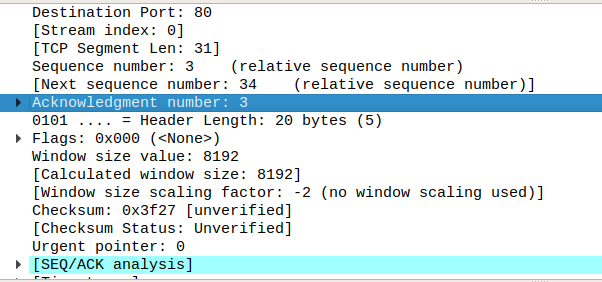


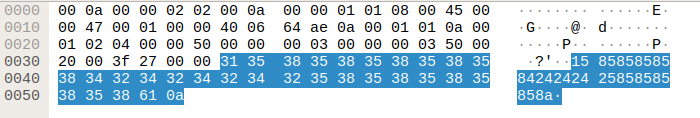
**Task 8 – Filter the tartget packet**

Use some filter command to filter all 116 datas.

To find what command we need, observe the text of each data.

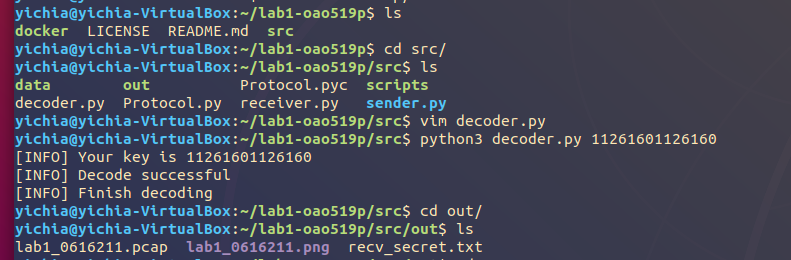
(both needed screenshots are at the top of this report)(Question part)

 Ex. Find the 14 datas with our secret key >> **TCP.ack ==3**

 Then find these 14 datas, with each contain a digit of secret key (first digit)

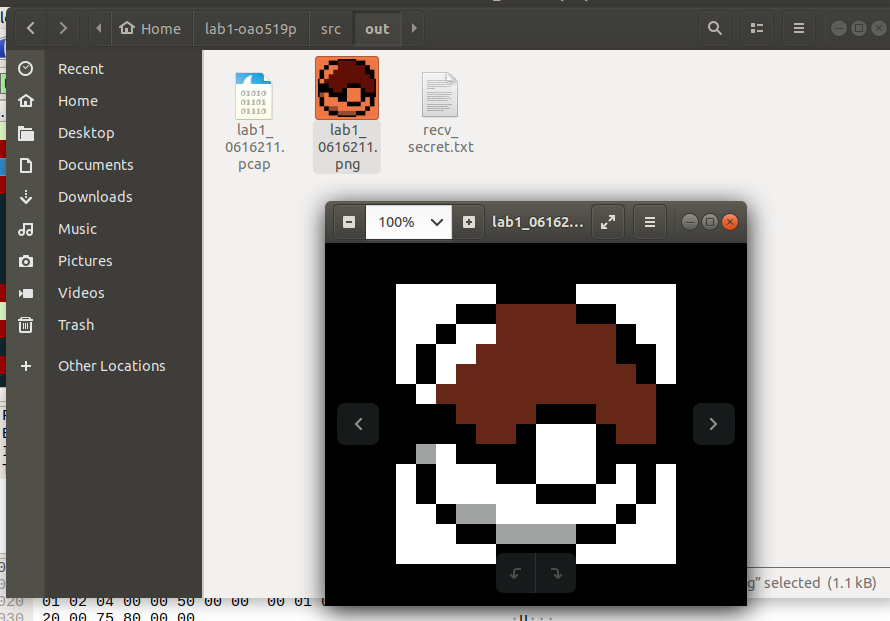
Get my secret key by combining them : **11261601126160**

**Task 9 – Decode the secret key**



First update our decoder.py (TA’s new version), then execute it.

Finally could see the image in ./src/out/



**Bonus**

1. **Learned in lab**: in lab, most I have learned actually are how to use commands (lol) It’s very cool as setting everything that can send and receive by left and right panes, and get the packets. (although most codes are not written by ourselves) By this lab, it gives me some kind of feeling about ‘’packet’’, and hope one day I could write all by myself.
2. **Difficulty have met:** in lab, most difficulty was due to unfamiliar. And can’t determine where the code has to be pasted, if the pasted place was wrong, it won’t get what we want, and error makes me confused which needs someone to answer me. (raising hand wasted lots of time) Task 1 cost most of my lab time, and luckily I didn’t meet any trouble at other Tasks, so the problem I met maybe can be solved by using these environments more.