NCTU CN2018 Lab. 1 – Packet Manipulation via Scapy

Student name: 楊郁欣 Student ID: 0616328 Department: CS

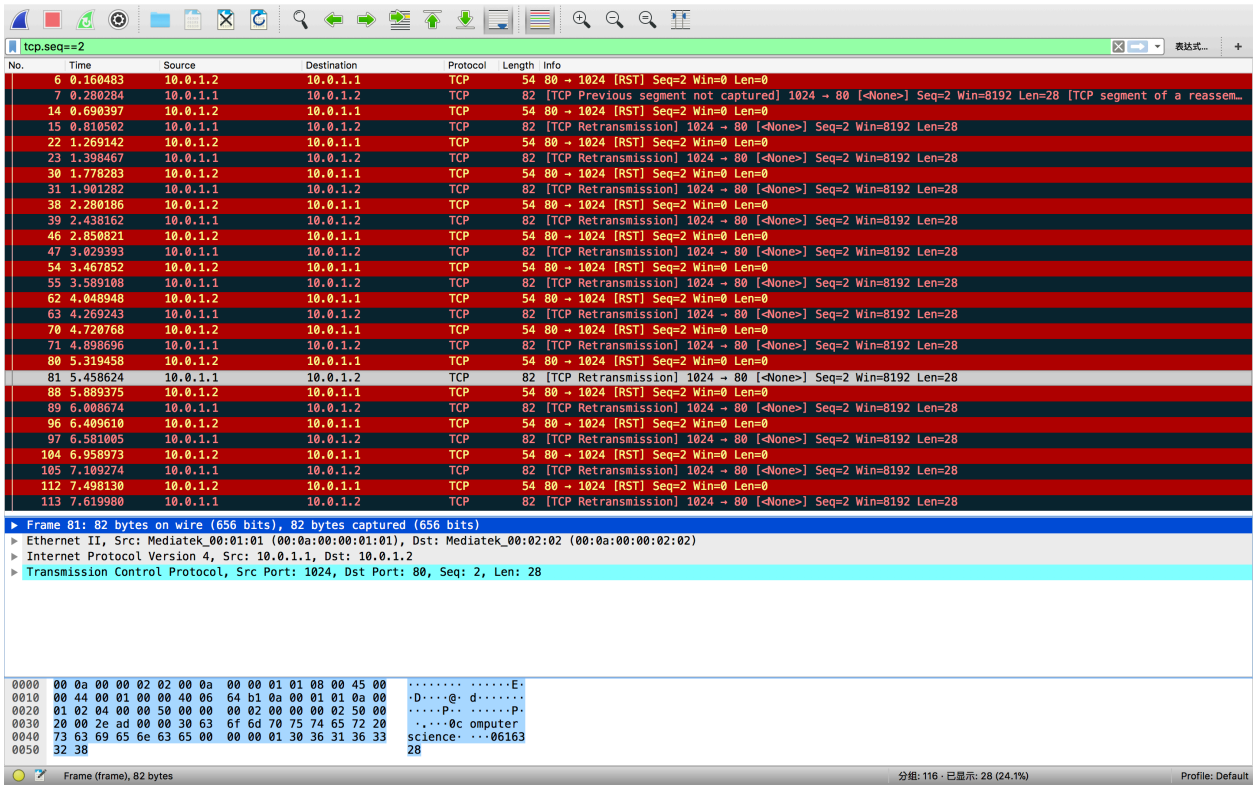
Part A. Questions

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

   Ans: tcp.seq == 2

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

   Ans:

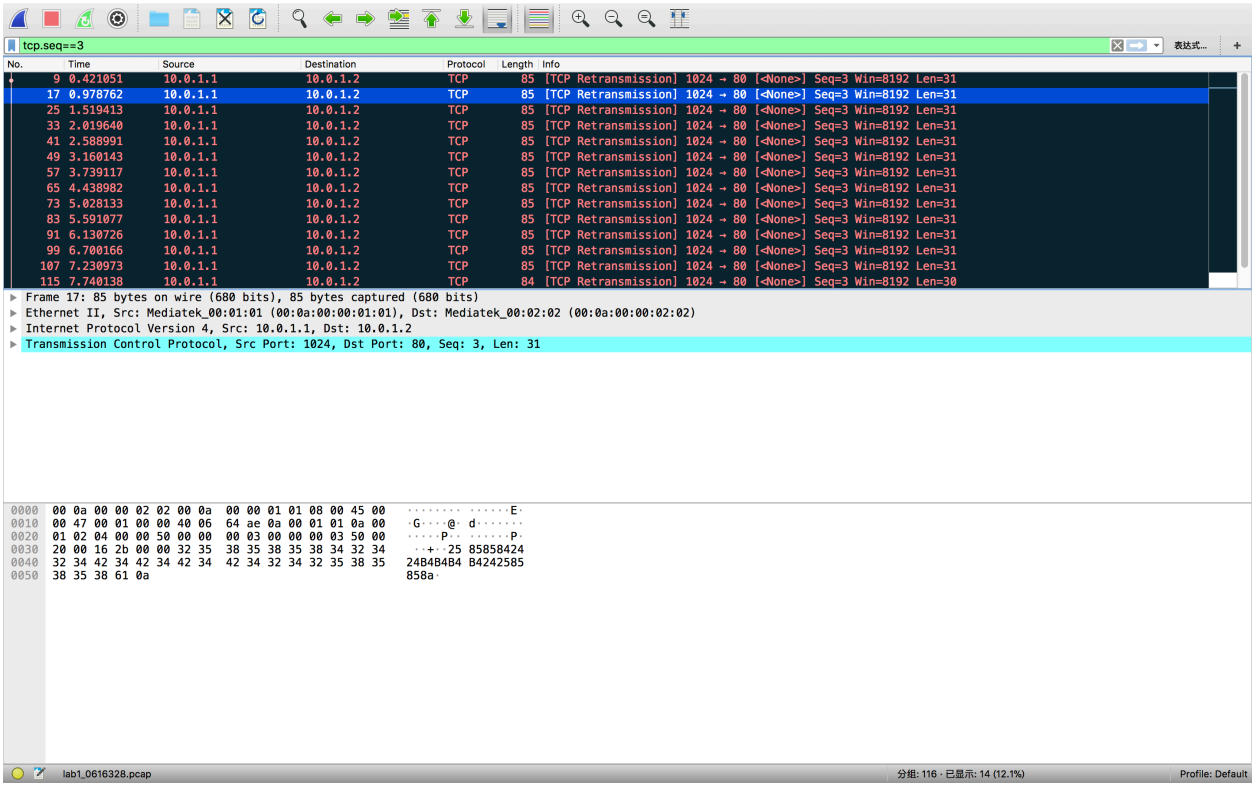


3. What is your command to filter the packet with “secret” payload on Wireshark?

   Ans: tcp.seq == 3

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

   Ans:



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

   Ans:

Part B. Description

Task 1 – Environment setup

• Configure Dockerfile

- How to configure Dockerfile?

  Ans: https://docs.docker.com/engine/reference/builder/

- Step 1:

  1. Download required files from GitHub

```
$ git clone
https://github.com/yungshenglu/Packet_Manipulation
```

  2. Get and set repository or global options

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

  <NAME> : sheeeep914

  <EMAIL> : cindy02017@gmail.com

  3. Set a new remote URL to your repository

```
$ git remote set-url origin
https://github.com/nctucn/lab1-<GITHUB_ID>.git
```

  <GITHUB_ID> : sheeeep914

  4. Push your repository to remote

```
$ git push origin master
```

- Step 2:

  Copy the following configuration to the Dockerfile

```
# Download base image from yungshenglu/ubuntu-env:16.04 (Task 1.)
FROM yungshenglu/ubuntu-env:16.04

# Update software respository (Task 1.)
RUN apt-get update
# Install software repository (Task 1.)
RUN apt-get install -y tcpdump

# Install pip packages (Task 1.)
RUN pip install scapy

# Set the container listens on the specified ports at runtime (Task 1.)
EXPOSE 22

# Clone the repository from GitHub (Task 1.)
RUN git clone https://github.com/yungshenglu/Packet_Manipulation.git
```

- Step 3:

  Open the Terminal and change the path to ./docker/

  and build the environment as follows:

```
$ sudo chmod +x main.sh
$ ./main.sh build cn2018 9487
```

• Login to **Docker** container using **SSH**

- Step 1:

  Use terminal to connect to the Docker

```
$ ssh root@0.0.0.0 -p 9487
Password: cn2018
```

- Step 2:

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

  (i.e., receiver)

```
# Create h2 network namespaces (Task 1.)
ip netns add h2
# Delete h2 network namespaces (Task 1.)
ip netns del h2
# Bring up the lookup interface in h2 (Task 1.)
ip netns exec h2 ip link set lo up
# Set the interface of h2 to h2-eth0 (Task 1.)
ip link set h2-eth0 netns h2
# Delete the interface of h2-eth0 (Task 1.)
ip link delete h2-eth0
# Activate h2-eth0 and assign IP address (Task 1.)
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
# Disable all IPv6 on h2-eth0 (Task 1.)
ip netns exec h2 sysctl net.ipv6.conf.h2-eth0.disable_ipv6=1
# Set the gateway of h2 to 10.0.1.254 (Task 1.)
ip netns exec h2 ip route add default via 10.0.1.254
```

- Step 3:

  Run main.sh to build the namespace

```
$ chmod +x main.sh
$ ./main.sh net
```

Task 2 – Define protocol via Scapy

- Define my protocol: **ID header format**
    - Step 1:

        Copy the following code to ./src/Protocol.py

        Use the format “Characters” in Python

```
class Protocol(Packet):
    # Set the name of protocol (Task 2.)
    name = 'Student'
    # Define the fields in protocol (Task 2.)
    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
    - Step 1:

```
# Set source and destination IP address (Task 3.)
src_ip = '10.0.1.1'
dst_ip = '10.0.1.2'

# Set source and destination port (Task 3.)
src_port = 1024
dst_port = 80

# Define IP header (Task 3.)
ip = IP(src = src_ip, dst = dst_ip)

# Define customized header (Task 3.)
my_id = '<YOUR_ID>'
my_dept = '<YOUR_DEPATMENT>'
my_gender = YOUR_GENDER
student = Protocol(id = my_id, dept = my_dept, gender =
my_gender)
```

my_id = ‘0616328'

my_dept = ‘computer science’

my_gender = ‘female’

- Send packets:
    - Step 1:

      Add the codes below in ./src/sender.py

```
# TCP connection - ACK (Task 3.)
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'

# Send packet with customized header (Task 3.)
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'
# Send packet with secret payload (Task 3.)
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:
    - Step 1:

        Add the codes below in ./src/receiver.py

```
# Set source IP address and destination interface (Task 4.)
dst_iface = 'h2-eth0'
src_ip = '10.0.1.1'

# Sniff packets on destination interface (Task 4.)
print '[INFO] Sniff on %s' % dst_iface
packets = sniff(iface = dst_iface, prn = lambda x:
packetHandler(x))

# Dump the sniffed packet into PCAP file (Task 4.)
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:

```
# Hint: Keep your path in ./src/
# Open tmux
$ tmux
# Open new pane in horizontal
Ctrl-b
Shift-%
# Switch between two panes
Ctrl-b
Arrow-left/right key
```

- Switch into two namespaces:

```
# Run namespace h1 in your left pane
$ ./scripts/main.sh run h1
# Run namespace h2 in your right pane
$ ./scripts/main.sh run h2
```

- Run receiver.py first:

```
# Switch between two panes
Ctrl-b
Arrow-right key
# Run receiver.py
h2> python receiver.py
```

- Run sender.py next:

```
# Switch between two panes
Ctrl-b
Arrow-left key
# Run sender.py
h1> python sender.py
```

- Use tcpdump to show my PCAP file

```
# Dump the PCAP via tcpdump
$ tcpdump -qns 0 -X -r <FILENAME>.pcap
```

<FILENAME> : lab1_0616328.pcap

(get a lab1_0616328.pcap and recv_secret.txt

after receiving all packets in ./src/out/)

Task 6 – Push your files to remote

- Push my image to Docker Hub

```
# Create a new image from a container’s changes
$ docker commit cn2018_c <DOCKER_HUB_ID>/cn2018_lab1
# Login to your Docker registry
$ docker login
# Push an image to a registry
$ docker push <DOCKER_HUB_ID>/cn2018_lab1
```

<DOCKER_HUB_ID> : sheeeep914

- Push my files to GitHub

```
# Get and set repository or global options
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
# Add your files into staging area
$ git add .
# Commit your files
$ git commit –m "Commit lab1 in class”
# Set the remote URL to your remote repository
$ git remote set-url origin
https://github.com/nctucn/lab1-<YOUR_ID>.git
# Push your files to remote repository
$ git push origin master
```

<NAME> : sheeeep914

<EMAIL> : cindy02017@gmail.com

<YOUR_ID> : sheeeep914

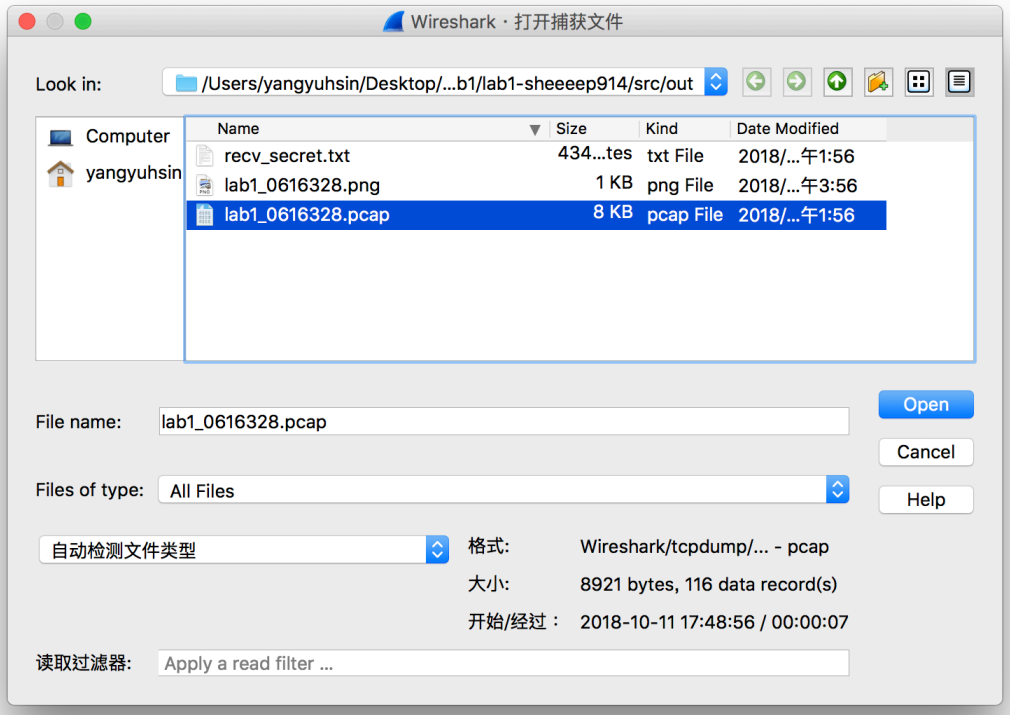
Task 7 – Load PCAP via Wireshark

- Download the code from GitHub

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

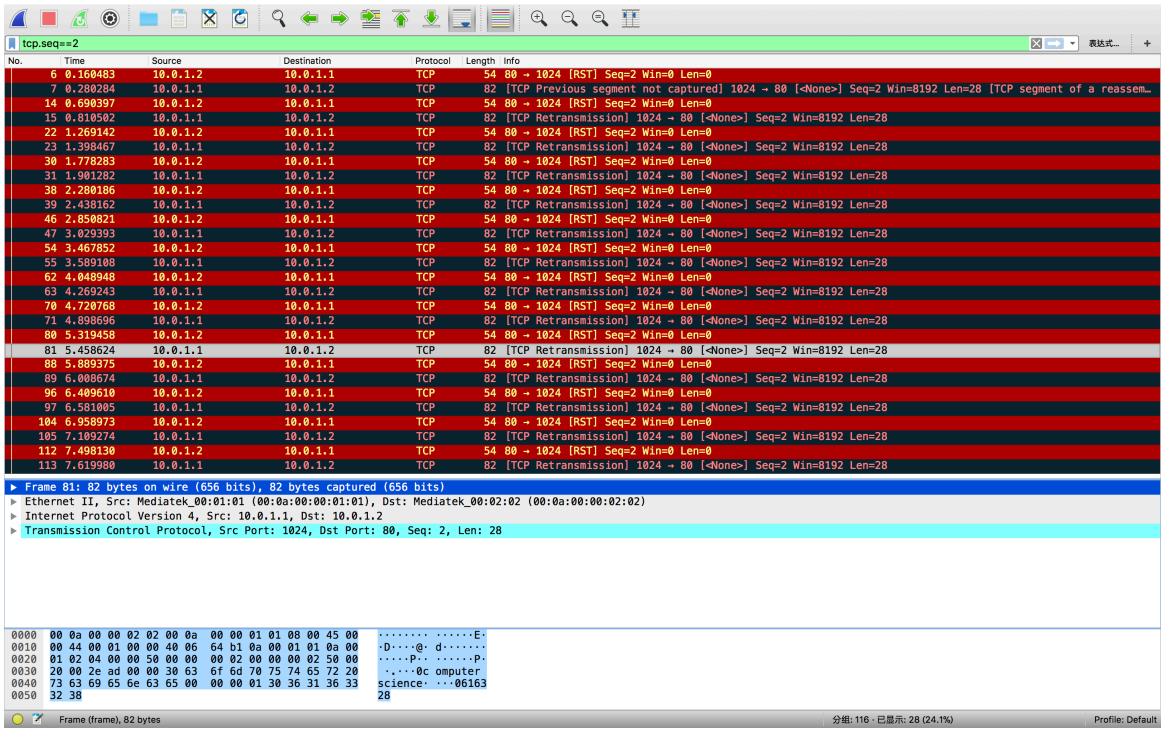
<YOUR_GITHUB_ID> : sheeeep914

- Open the PCAP file using Wireshark

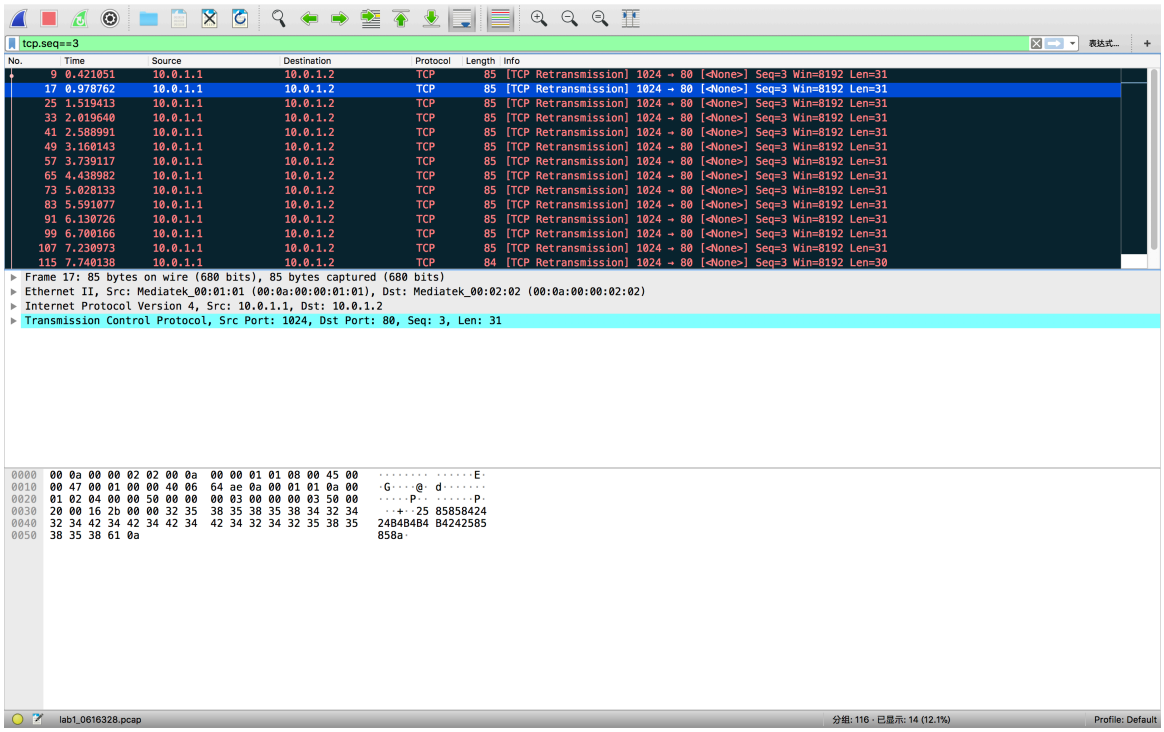


Task 8 – Filter the target packet

- Filter the packets of our defined protocol
  - Filter rule: tcp.seq == 2



- Filter the packets with the “secret” bits
  - Filter rule: tcp.seq == 3



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

  Ans: (1) My secret key: 82361608236160

  (2) Find the first digit in a “secret” payload, and combine into 14 digits from 14 secret packets.

Task 9 – Decode the secret key

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

```
python decoder.py 82361608236160
```

  - My result is?