Password

- Windows
  - Account: **course-icn**
  - Password: **cscs**

Computer Networks
@CS.NCTU

Lab. 1: Packet Manipulation via Scapy

Lab. Tasks

Location: EC-315, 316

Instructor: 陸勇盛 (David Lu)

Agenda

- Overview
- Objectives
- Installation
- Tasks
- Submission
- Grading Policy

Objectives

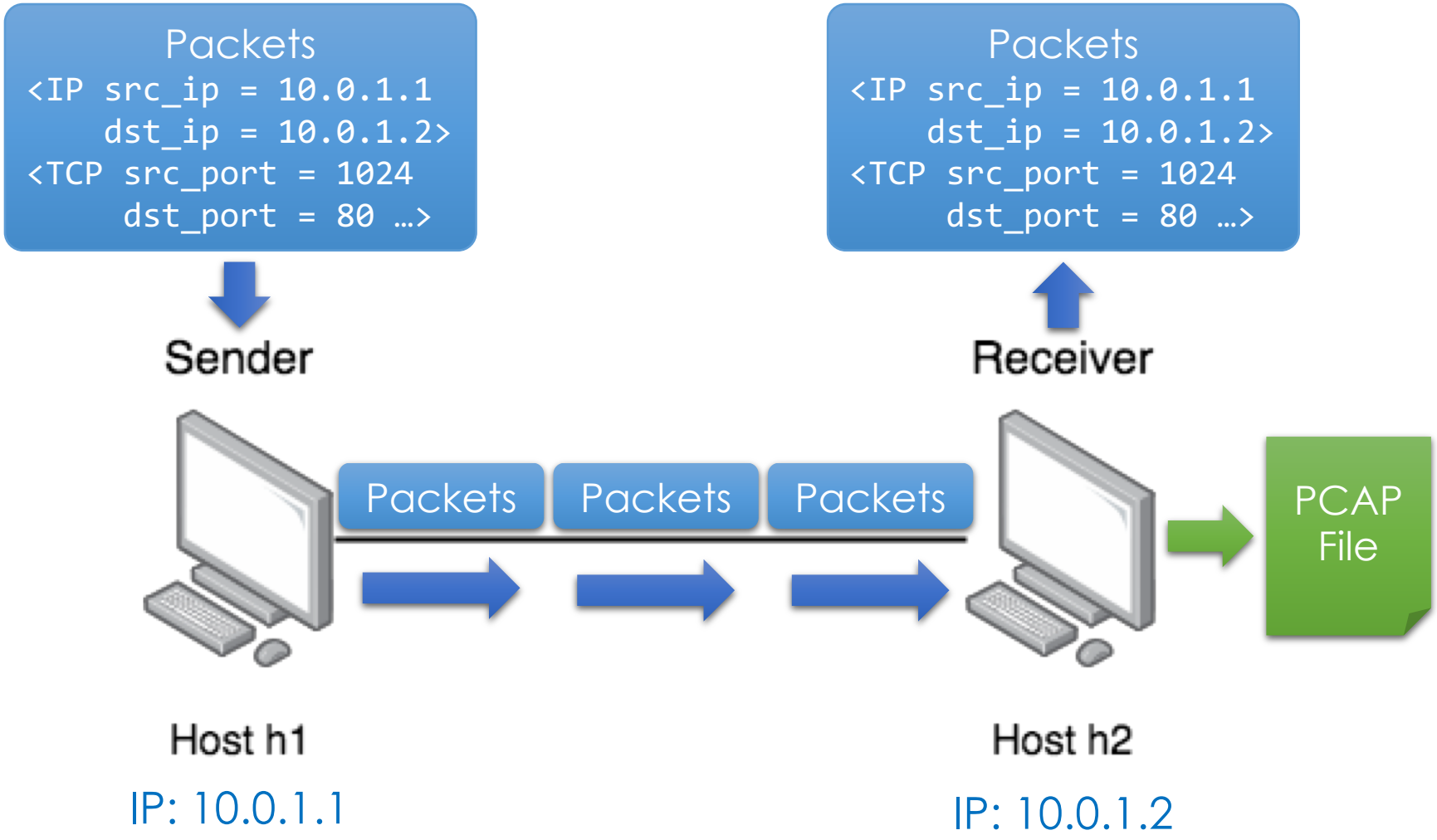
1. Learn how to define your own protocol and generate a packet payload
2. Learn how to use Wireshark to filter packets and find your wanted information

Minor Objectives

This lab aims to learn how we use Scapy and Python to program a simple network protocol and observe the behavior of packet sending and receiving via Wireshark

- Basic knowledge of Docker
- Linux networking
- Python with Scapy
- Wireshark

Overview


Packets
<IP src_ip = 10.0.1.1
dst_ip = 10.0.1.2>
<TCP src_port = 1024
dst_port = 80 …>
Packets
<IP src_ip = 10.0.1.1
dst_ip = 10.0.1.2>
<TCP src_port = 1024
dst_port = 80 …>
Sender
Receiver
Packets
Packets
Packets
PCAP File
Host h1
IP: 10.0.1.1
Host h2
IP: 10.0.1.2

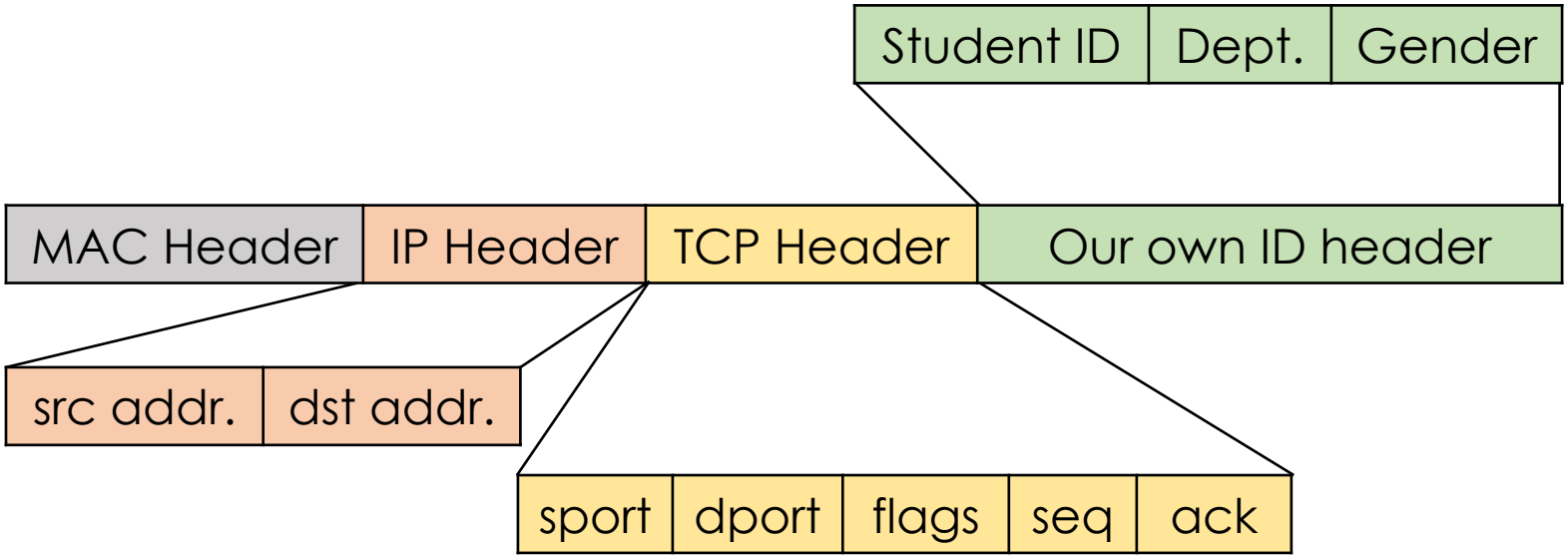
Overview (cont.)

- Define our own proprietary protocol
- In this protocol, we will iteratively send to a server
  1. **ID packet**:
     your (ID + department + gender)
  2. **Secret packet**:
     a digit of the secret key
- The above procedure will repeat 14 times so that you will collect a 14-digit secret key
  - E.g., 41228904512480

ID packet 1
Secret packet 1
ID packet 2
Secret packet 2
.
.
.
ID packet 14
Secret packet 14

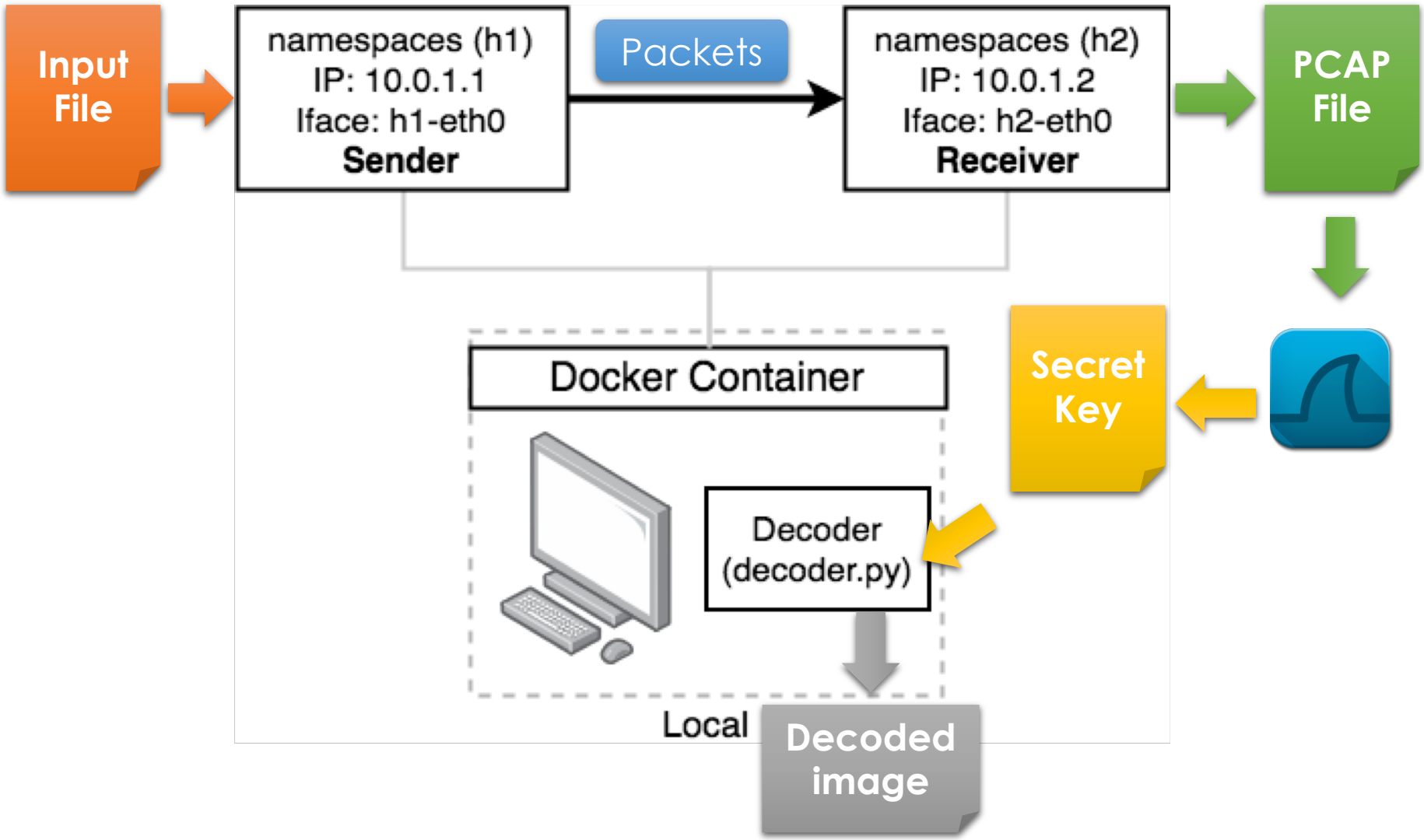
Overview (cont.)

- Packet format
  - **ID packet**



  - **Secret packet**

Overview (cont.)



Installation

- **Docker (Docker CE)**
  - Windows / MacOS / Ubuntu Linux / Others
  - Please register an account on Docker Hub

- **Wireshark 2.6.3**
  - Windows (32-bit / 64-bit) / MacOS
  - Ubuntu Linux

```
$ sudo apt-get install wireshark
```

- **Others**
  - PieTTY (for Windows)

Tasks

- **In lab assignment (Oct. 11 @EC-315, 316)**
  1. Environment Setup
  2. Define protocol via Scapy
  3. Send packets
  4. Sniff packets
  5. Run sender and receiver
  6. Push your files to remote
- **Homework assignment**
  7. Load PCAP via Wireshark
  8. Filter the target packet
  9. Decode the secret key
  10. Report

Tasks (In Class)

1. Environment setup
2. Define protocol via Scapy
3. Send packets
4. Sniff packets
5. Run sender and receiver
6. Push your files to remote

Task 1. Environment setup

- Download required files from GitHub

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

- Get and set repository or global options

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

- Set a new remote URL to your repository

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

- Push your repository to remote

```
$ git push origin master
```

Task 1. Environment setup (cont.)

Structure of the packet manipulation project

```
Packet Manipulation/                # This is ./ in this repository
|--- docker/                        # Docker configuration
     |--- Dockerfile
     |--- main.sh                   # Scripts for running Docker
     |--- [Other files…]
|--- src/                           # Source code
     |--- data/                     # Input files
          |--- record.txt           # Example file for R/W
     |--- out/                      # Output files
     |--- scripts/                  # Networks configuration
          |--- main.sh              # Scripts for build namespace
          |--- [Other files…]
     |--- sender.py                 # Send packets
     |--- receiver.py               # Receive and sniff packets
     |--- Protocol.py               # Define your own protocol
     |--- decoder.py                # Decode the output file
|--- LICENSE
|--- README.md
```

Task 1. Environment setup (cont.)

- Copy the following configuration to the Dockerfile (./docker/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
```

Task 1. Environment setup (cont.)

- **For windows**
  - Open the CMD (administration) and change the path to ./docker/ and build the environment as follows:

```
# Build the image from Dockerfile
docker build -t cn2018 .
# Build a container named cn2018_c from cn2018
docker run -d -p 9487:22 --privileged --name cn2018_c
cn2018
# List port 22 mapping on cn2018_c
docker port cn2018_c 22
```

- **For MacOS and Ubuntu**
  - 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
```

Task 1. Environment setup (cont.)

- You will get the following result if succeed

```
[INFO] Docker image: cn2018
[INFO] External port: 9487
Sending build context to Docker daemon  9.216kB
Step 1/12 : FROM yungshenglu/ubuntu-env:16.04
 ---> 3d2d9b1b0c9d
Step 2/12 : RUN apt-get update
 ---> Using cache
 ---> d34308efb7ce
Step 3/12 : RUN apt-get install -y tcpdump
 ---> Using cache
 ---> 78d899c28245
Step 4/12 : RUN pip install scapy
 ---> Using cache
 ---> cca3b629034a
Step 5/12 : RUN echo 'root:cn2018' | chpasswd
 ---> Using cache
 ---> 4be6592ac9e4
Step 6/12 : RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
 ---> Using cache
 ---> eb9d6782d52a
Step 7/12 : RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd
 ---> Using cache
 ---> ee333a96a800
Step 8/12 : ENV NOTVISIBLE "in users profile"
 ---> Using cache
 ---> 710173f72874
Step 9/12 : ENV LC_ALL C
 ---> Using cache
 ---> b3cc7ea636d1
Step 10/12 : RUN echo "export VISIBLE=now" >> /etc/profile
 ---> Using cache
 ---> 2af02906b668
Step 11/12 : EXPOSE 22
 ---> Using cache
 ---> a3eec5b1837e
Step 12/12 : CMD ["/usr/sbin/sshd", "-D"]
 ---> Using cache
 ---> d9e4ee8987da
Successfully built d9e4ee8987da
Successfully tagged cn2018:latest
0.0.0.0:9487
```

Task 1. Environment setup (cont.)

- Troubleshooting if you see errors

```
Successfully built d9e4ee8987da
Successfully tagged cn2018:latest
docker: Error response from daemon: Conflict. The container name
"/cn2018_c" is already in use by container
"1fc7ecfe83d5265ec76b47954501404079 5d0335fd599650bf6f593178ea1c4b".
You have to remove (or rename) that container to be able to reuse
that name.
See 'docker run --help’.

# Stop the container that has already named cn2018_c
$ docker container stop
1fc7ecfe83d5265ec76b47954501404079 5d0335fd599650bf6f593178ea1c4b
# Remove the container named cn2018_c
$ docker container rm
1fc7ecfe83d5265ec76b47954501404079 5d0335fd599650bf6f593178ea1c4b
# Re-build the container again (follow the slide 12)
```

Task 1. Environment setup (cont.)

- Login to your **Docker** container using **SSH**
- **For windows**
  - Open the PieTTY and connect to the Docker
    - IP address: `127.0.0.1`
    - Port: `9487`
  - Login as root

```
Login: root
Password: cn2018
```

- **For MacOS and Ubuntu**
  - Use terminal to connect to the Docker

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

Task 1. Environment setup (cont.)

- Create the namespace in ./src/scripts/main.sh for h2 (i.e., receiver) (h1's namespace has been created)

```
# 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
```

Task 1. Environment setup (cont.)

- Run main.sh to build the namespace

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

- You will get the following result if succeed

```
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
[INFO] Create h1 and h2 network namespaces
[INFO] Bring up the lookup interface in h1 and h2
[INFO] Build the link: h1-eth0 <-> h2-eth0
[INFO] Activate h1-eth0 and assign IP address
[INFO] Activate h2-eth0 and assign IP address
[INFO] Disable all IPv6 on h1-eth0 and h2-eth0
net.ipv6.conf.h1-eth0.disable_ipv6 = 1
net.ipv6.conf.h2-eth0.disable_ipv6 = 1
[INFO] Set the gateway to 10.0.1.254 in routing table
```

Task 2. Define protocol via Scapy

- Define your protocol: **ID header format**
  - Copy the following code to ./src/Protocol.py

```
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),
    ]
```

  - Use the format "Characters" in Python

Task 3. Send packets

- Set your own packet header in ./src/sender.py

```
# 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)
```

Task 3. Send packets (cont.)

- Send packets:
  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'
```

Task 3. Send packets (cont.)

- Send packets:
  Add the codes below in ./src/sender.py

```
# 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:
  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
```

Task 5. Run sender and receiver

- Run receiver.py first

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

- Run sender.py

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

Task 5. Run sender and receiver

- Use tcpdump to show your PCAP file

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

- You will get a lab1_ID.pcap and recv_secret.txt after receiving all packets in ./src/out/
  - **Note: this is the file required for homework**

Task 6. Push your files to remote

- Push your 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
```

Task 6. Push your files to remote

- Push your 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
```

Tasks (Homework)

7. Load the packet trace via Wireshark
8. Filter the target packets
9. Decode the secret key
10. Report

Task 7. Load PCAP via Wireshark

- Download your code from GitHub

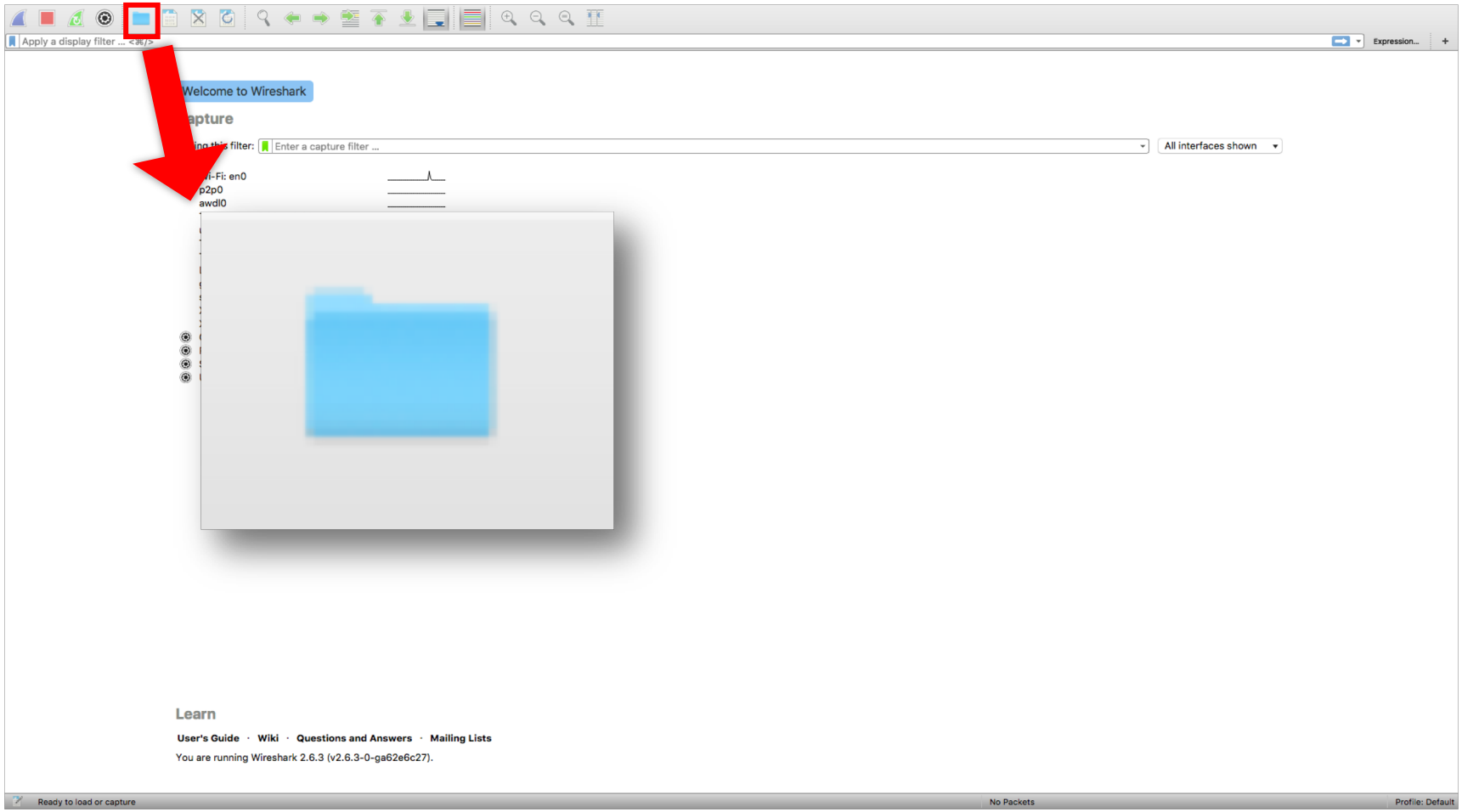
```
$ git clone https://github.com/nctucn/lab1-
<YOUR_GITHUB_ID>.git
```

- Install **Wireshark 2.6.3**
  - Windows (32-bit / 64-bit) / MacOS
  - Ubuntu Linux

```
$ sudo apt-get install wireshark
```

Task 7. Load PCAP via Wireshark

- **Open the PCAP file using Wireshark** (./src/out/lab1_yourID.pcap)



Task 8. Filter the target packet

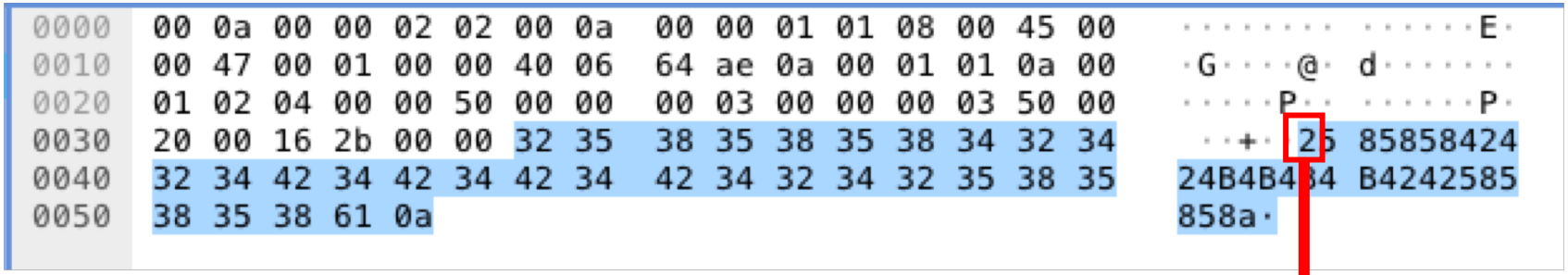
- **Filter the packets of our defined protocol**
  - Enter your filter command on DisplayFilters
  - Hint: use header info., e.g., IP address or/and TCP seq

```
>> tcp.port eq 25 or icmp  # Example command
```

  - Save the screenshot of your filtering result
- **Filter the packets with the "secret" bits**
  - Enter your filter command on DisplayFilters
  - Find out the first digit of the "secret" payload in these packets and combine them as a 14-digit "secret" key
  - Save the screenshot of your filtering result
- **Notice:** insert the above two screenshots into your report (no need to output any files)

Task 8. Filter the target packet

- Example of getting the first digit in a "secret" packet



Combine into 14 digits from 14 secret packets

The first digit in a "secret" payload

Task 9. Decode the secret key

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

```
$ python decoder.py <YOUR_SECRET_KEY>
```

    - **Do NOT modify any code in decoder.py!**
  - The output file is in ./src/out/
    - You will get an image related to Pokemon if succeed

Task 10. Report

- **Answer the following questions in short**
  1. What is your command to filter the packet with customized header on Wireshark?
  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?
  4. Show the screenshot of filtering the packet with "secret" payload .
  5. Show the result after decoding the "secret" payload.

Task 10. Report (cont.)

- **Describe each step in this lab in detail**
  - e.g., which bottoms you click, what are the filtering rules, etc.
- **Bonus**
  You should answer the following all two questions; otherwise, you will not get bonus points.
  - What you have learned in this lab?
  - What difficulty you have met in this lab?

Submission

- **Submit your works to GitHub repository**

```
# Add all files into staging area
$ git add .
# Commit your files
$ git commit -m "YOUR OWN COMMIT MESSAGE"
# Push your files to remote
$ git push origin master
```

- **Notice**
  - You should not commit your works only one time; otherwise, your lab may be deemed as plagiarism
  - You should write your commit message clearly
    - How to Write a Git Commit Message
  - Make sure that your final work is on `master` branch

Submission (cont.)

- **Trace files** (./src/out/)
  - PCAP file (lab1_ID.pcap)
  - Decoded image (lab1_ID.png)
- **Python code** (./src/)
  - sender.py
  - receiver.py
  - Protocol.py

Grading Policy

- **Deadline - Oct. 25, 2018. 23:00**
- **In lab course (Oct. 11), 55%**
  1. Environment setup, 10%
  2. Define protocol via Scapy, 10%
  3. Send packets, 10%
  4. Sniff packets, 10%
  5. Run sender and receiver, 10%
  6. Push your files to remote, 5%
- **Homework, 45%**
  1. PCAP file, 10%
  2. Decoding result, 10%
  3. Report, 25% (Bonus 5%)

Grading Policy (cont.)

- **Late Policy** (follow syllabus)

$$(\textbf{Your score}) \times 0.8^{D},$$

where D is the number of days over due

- **Cheating Policy** (follow syllabus)
  - Academic integrity
  - Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater will be held responsible for the cheating

Checking Rules in Lab Course

- Each tasks (in lab course) you have 3 chances to call TA to check whether you pass or not
- If you pass within 3 chances, you will get all score of the task.
  - E.g., Task 1. Environment setup, 10%
    - If you pass within 3 chances, you will get all 10%
    - If you pass in the 4th chances, you will get 9%
    - If you pass in the 5th chances, you will get 8%
    - etc.
- If you have any question, you can ask TA and without counting into 3 chances.