Computer Security Capstone

Project II: MITM and Pharming Attacks in Wi-Fi Networks

Chi-Yu Li (2021 Spring)
Computer Science Department
National Yang Ming Chiao Tung University

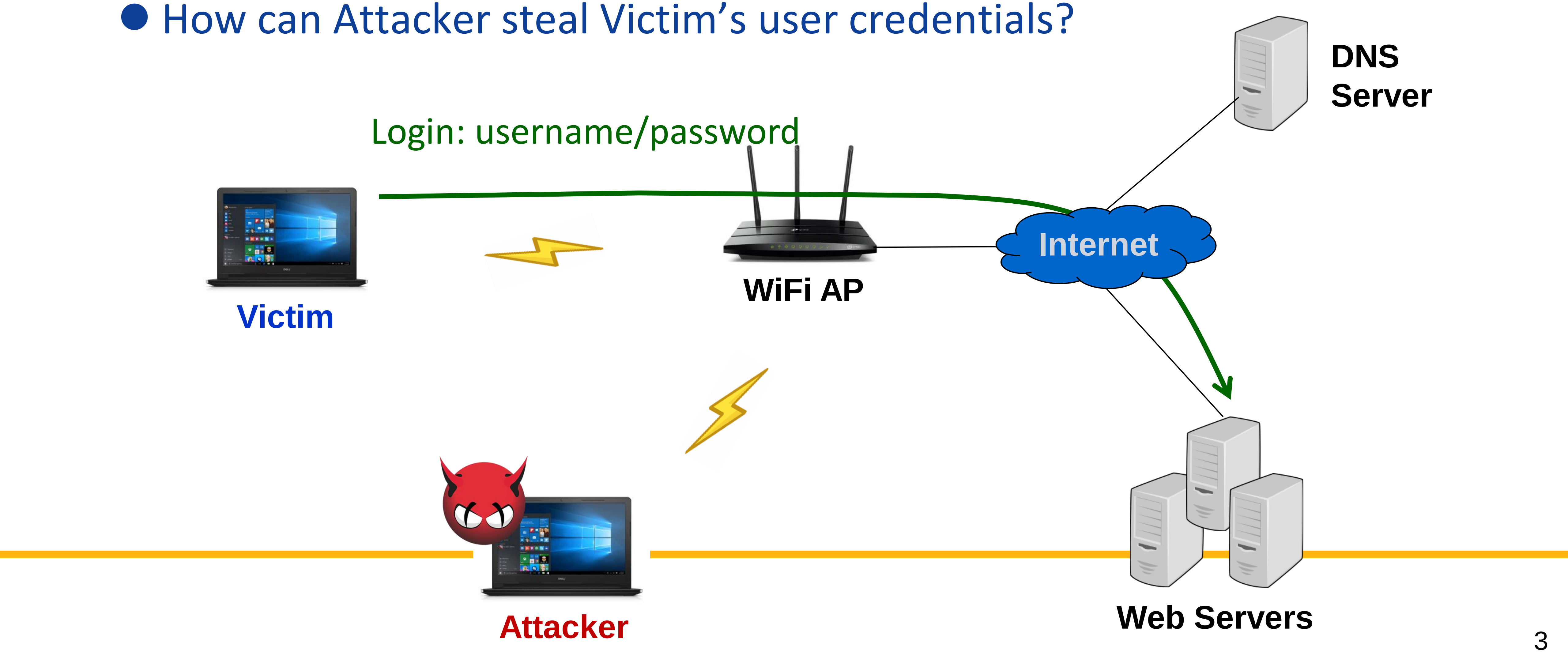
Goal

- Understand how user credentials can be leaked by a man-in-the-middle (MITM) attack over Wi-Fi networks

- You will learn how to
  - □ scan IP/MAC addresses of the devices in a Wi-Fi network
  - □ launch an ARP spoofing attack
  - □ launch a man-in-the-middle attack
  - □ launch a pharming attack

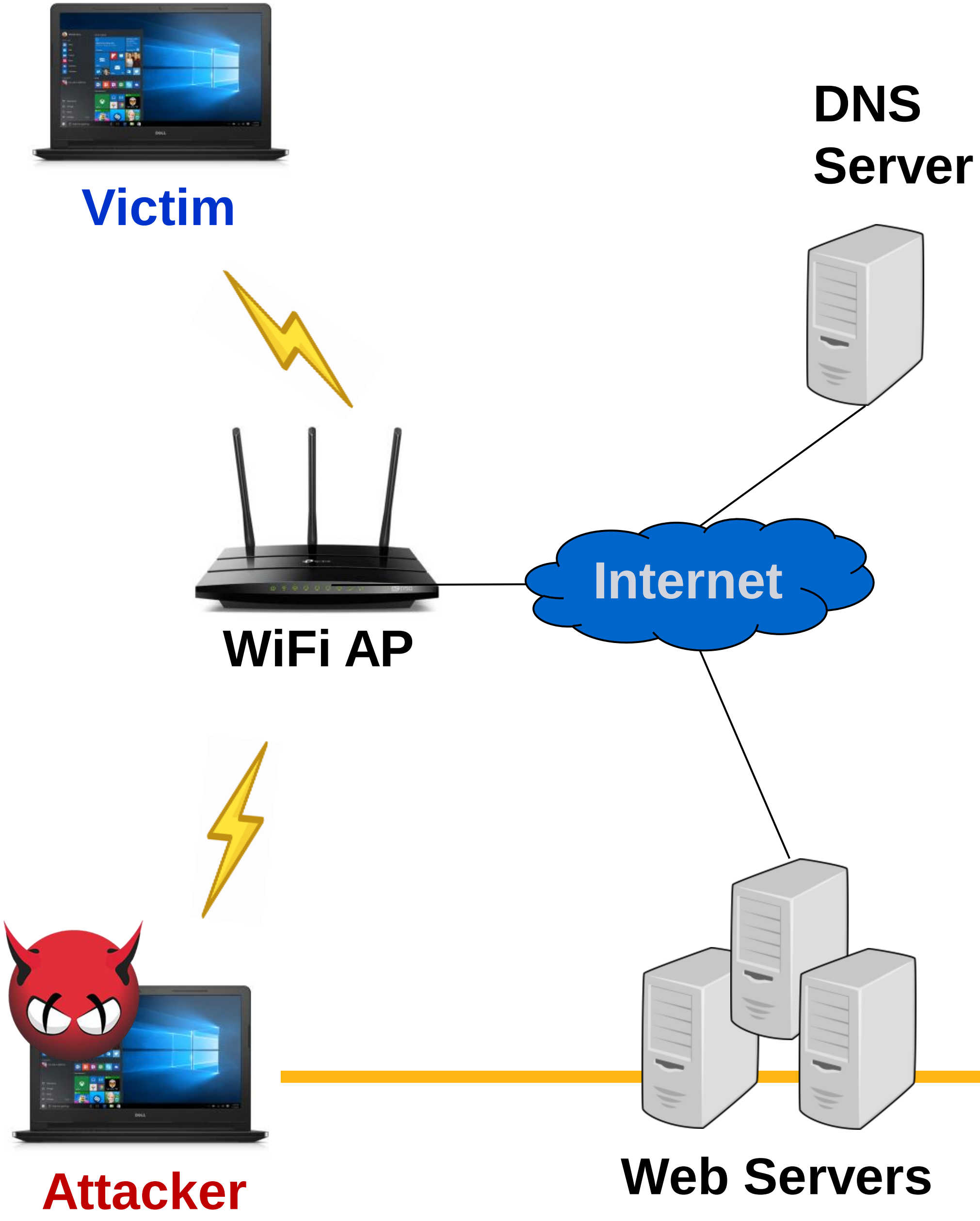
Attack Scenario

- How can Attacker steal Victim's user credentials?



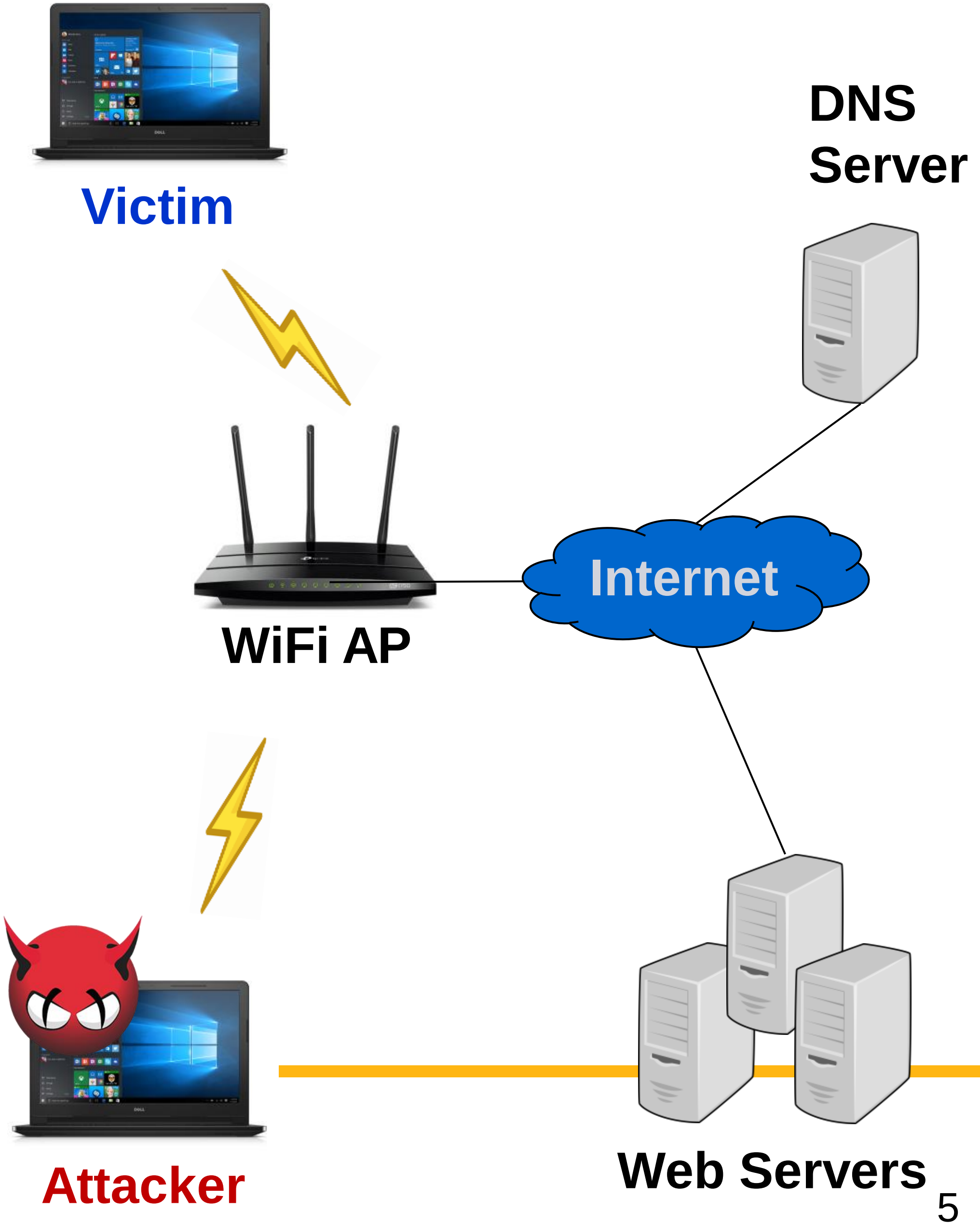
Major Ideas

- Redirect Victim’s traffic to Attacker
  - □ Man-in-the-middle based on ARP spoofing
  - □ How to know Victim’s IP/MAC address?
- How about encrypted sessions?
  - □ MITM attack: split the encrypted sessions
  - □ Pharming attack: redirect HTTP requests to a phishing web page



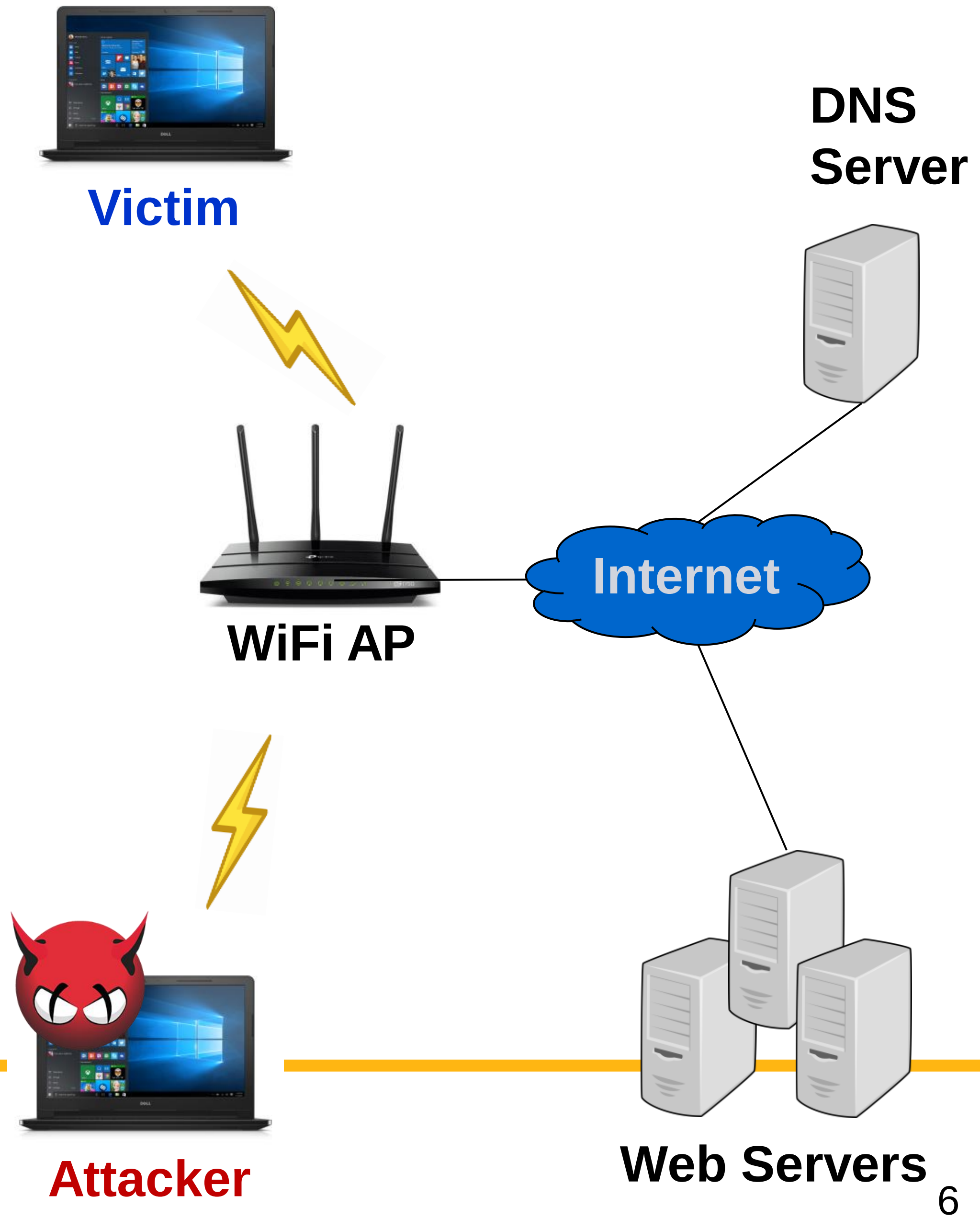
Tasks: MITM and Pharming

- MITM Attack (50%)
  - □ Obtain all other client devices' IP/MAC addresses in a connected Wi-Fi network (Task I: 20%)
  - □ ARP spoofing for all other client devices in the Wi-Fi network (Task II: 15%)
  - □ Split SSL/TLS encrypted sessions and get the inputted username/password strings from HTTPS sessions (Task III: 15%)



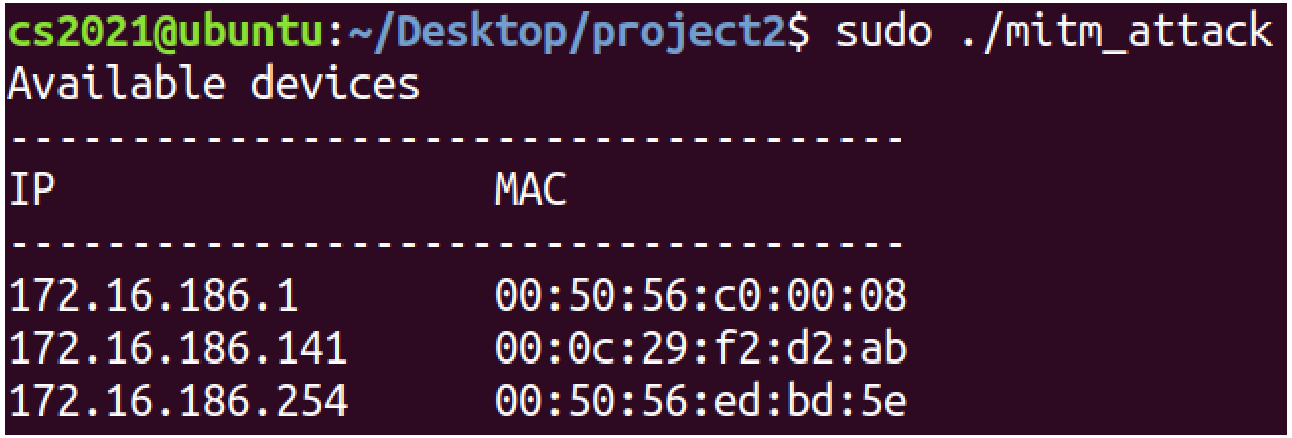
Tasks: MITM and Pharming

- Pharming Attack (30%)
  - □ Obtain all other client devices' IP/MAC addresses in a connected Wi-Fi network
  - □ DNS spoofing attack for web services (Task IV: 30%)



Task I: Device Address Information Collection

- Scan all the devices' IP/MAC addresses in the Wi-Fi network
  - □ You can use 'scapy' and 'netifaces' library in Python or commands 'nmap', 'arp', and 'route'



```
cs2021@ubuntu:~/Desktop/project2$ sudo ./mitm_attack
Available devices
-------------------------------------
IP                  MAC
-------------------------------------
172.16.186.1        00:50:56:c0:00:08
172.16.186.141      00:0c:29:f2:d2:ab
172.16.186.254      00:50:56:ed:bd:5e
```

- Fetch the IP/MAC addresses of all the other client devices

Task II: ARP Spoofing

- What is ARP (Address Translation Protocol)?
  - A communication protocol: discovering the link layer (or MAC) address associated with a given IP
  - A request-response protocol: messages are encapsulated by a link-layer protocol
    - ARP request: broadcast
    - ARP response: unicast
  - Never routed across internetworking nodes

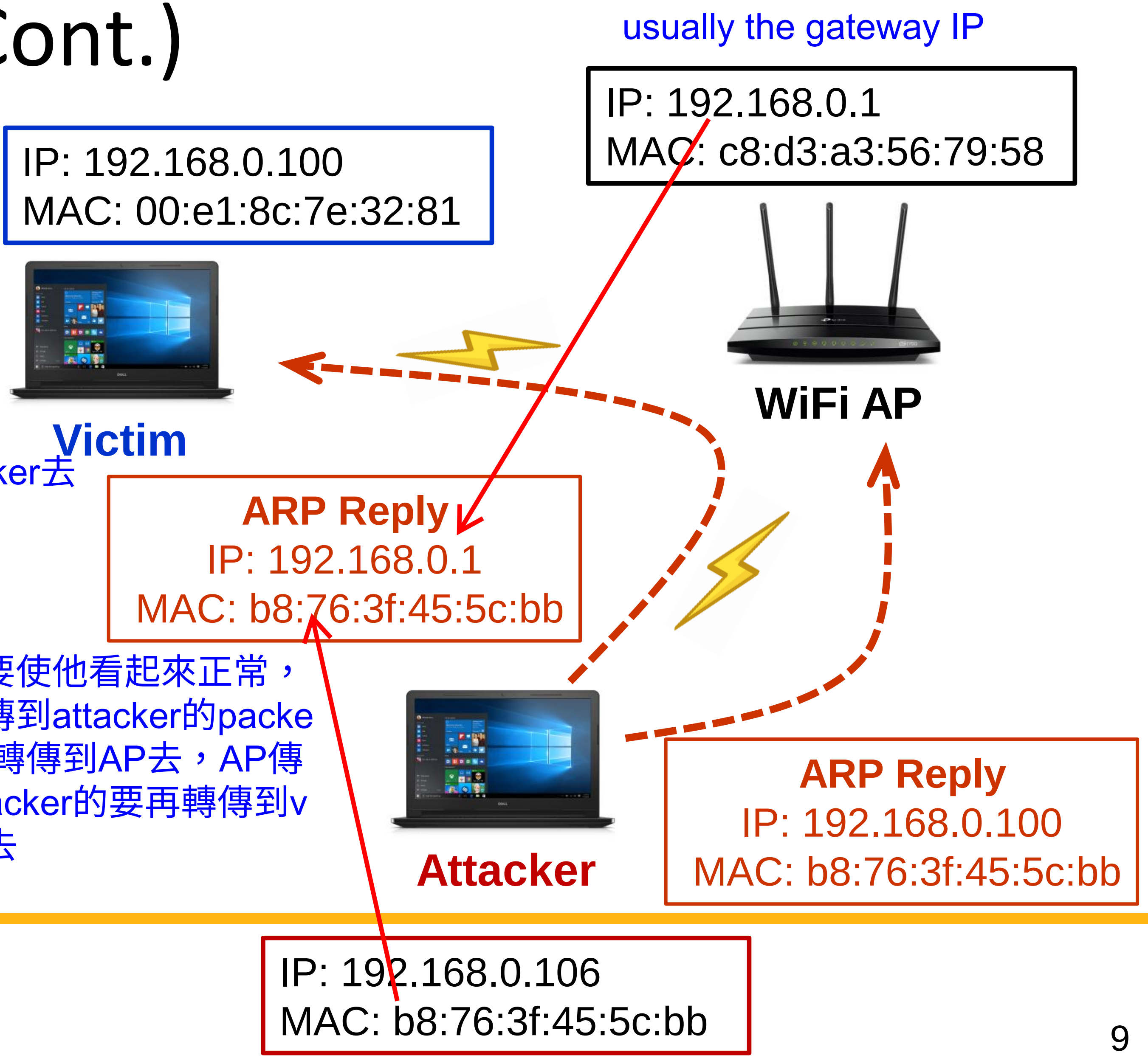
Task II: ARP Spoofing (Cont.)

- Generate spoofed ARP replies for all other client devices
  - □ You can use ‘scapy’ library in Python

所以victim傳出去的資料會到attacker去
因為wifi傳輸是經由MAC Layer

- Both uplink and downlink should be considered
  - □ Other client devices’ network services can work normally

為了要使他看起來正常，我們傳到attacker的packet要再轉傳到AP去，AP傳到attacker的要再轉傳到victim去



usually the gateway IP

IP: 192.168.0.1
MAC: c8:d3:a3:56:79:58

IP: 192.168.0.100
MAC: 00:e1:8c:7e:32:81

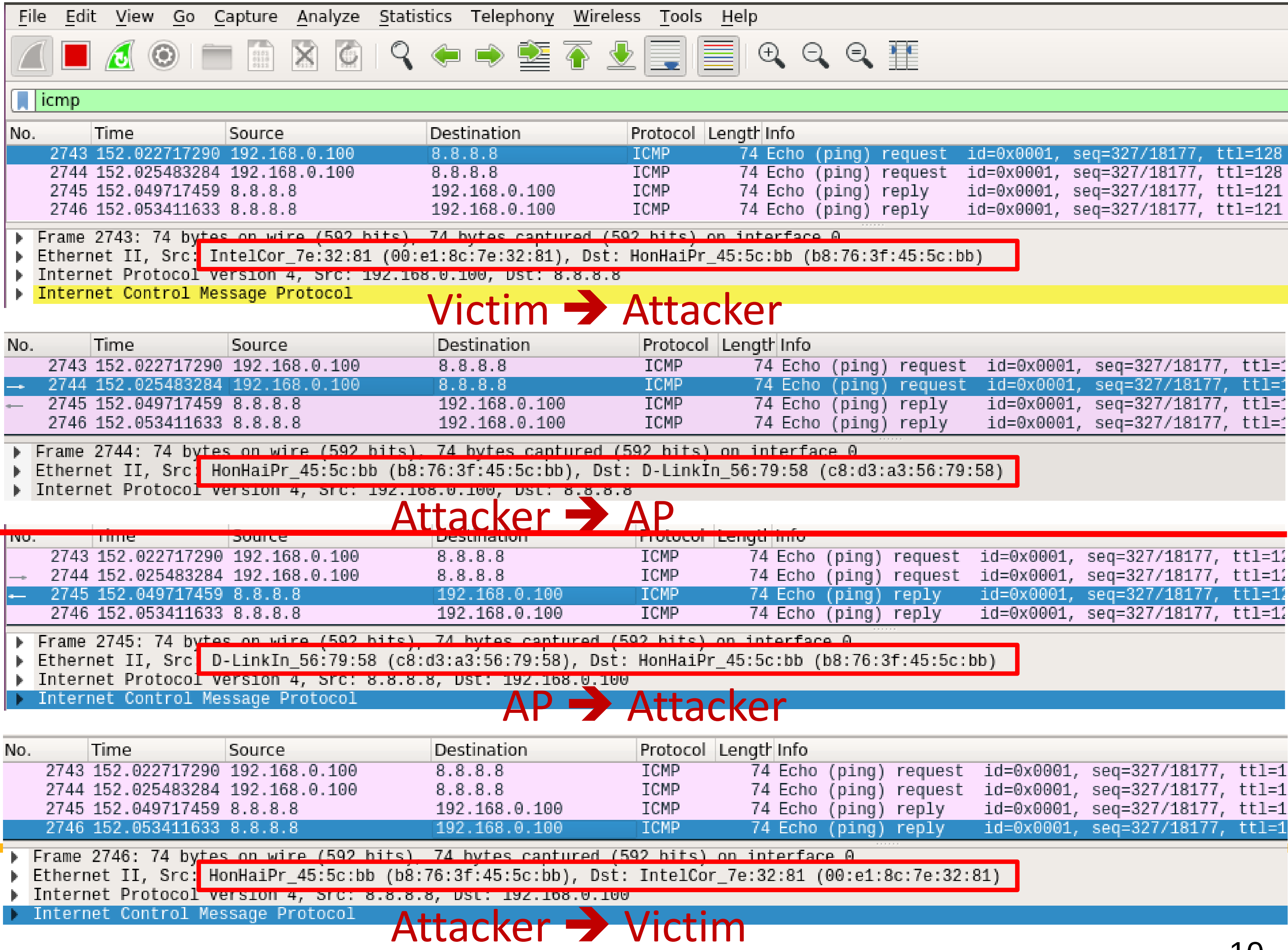
ARP Reply
IP: 192.168.0.1
MAC: b8:76:3f:45:5c:bb

ARP Reply
IP: 192.168.0.100
MAC: b8:76:3f:45:5c:bb

IP: 192.168.0.106
MAC: b8:76:3f:45:5c:bb

Task II: ARP Spoofing (Cont.)

- An example trace of the successful ARP spoofing at Attacker


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

```
No.   Time            Source          Destination     Protocol Length Info
 2743 152.022717290 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=128
 2744 152.025483284 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=128
 2745 152.049717459 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=121
 2746 152.053411633 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=121
▸ Frame 2743: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: IntelCor_7e:32:81 (00:e1:8c:7e:32:81), Dst: HonHaiPr_45:5c:bb (b8:76:3f:45:5c:bb)
▸ Internet Protocol Version 4, Src: 192.168.0.100, Dst: 8.8.8.8
▸ Internet Control Message Protocol
```

Victim ➔ Attacker

```
No.   Time            Source          Destination     Protocol Length Info
 2743 152.022717290 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=1
→2744 152.025483284 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=1
←2745 152.049717459 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=1
 2746 152.053411633 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=1
▸ Frame 2744: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_45:5c:bb (b8:76:3f:45:5c:bb), Dst: D-LinkIn_56:79:58 (c8:d3:a3:56:79:58)
▸ Internet Protocol Version 4, Src: 192.168.0.100, Dst: 8.8.8.8
```

Attacker ➔ AP

```
No.   Time            Source          Destination     Protocol Length Info
 2743 152.022717290 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=12
→2744 152.025483284 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=12
←2745 152.049717459 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=12
 2746 152.053411633 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=12
▸ Frame 2745: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: D-LinkIn_56:79:58 (c8:d3:a3:56:79:58), Dst: HonHaiPr_45:5c:bb (b8:76:3f:45:5c:bb)
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.0.100
▸ Internet Control Message Protocol
```

AP ➔ Attacker

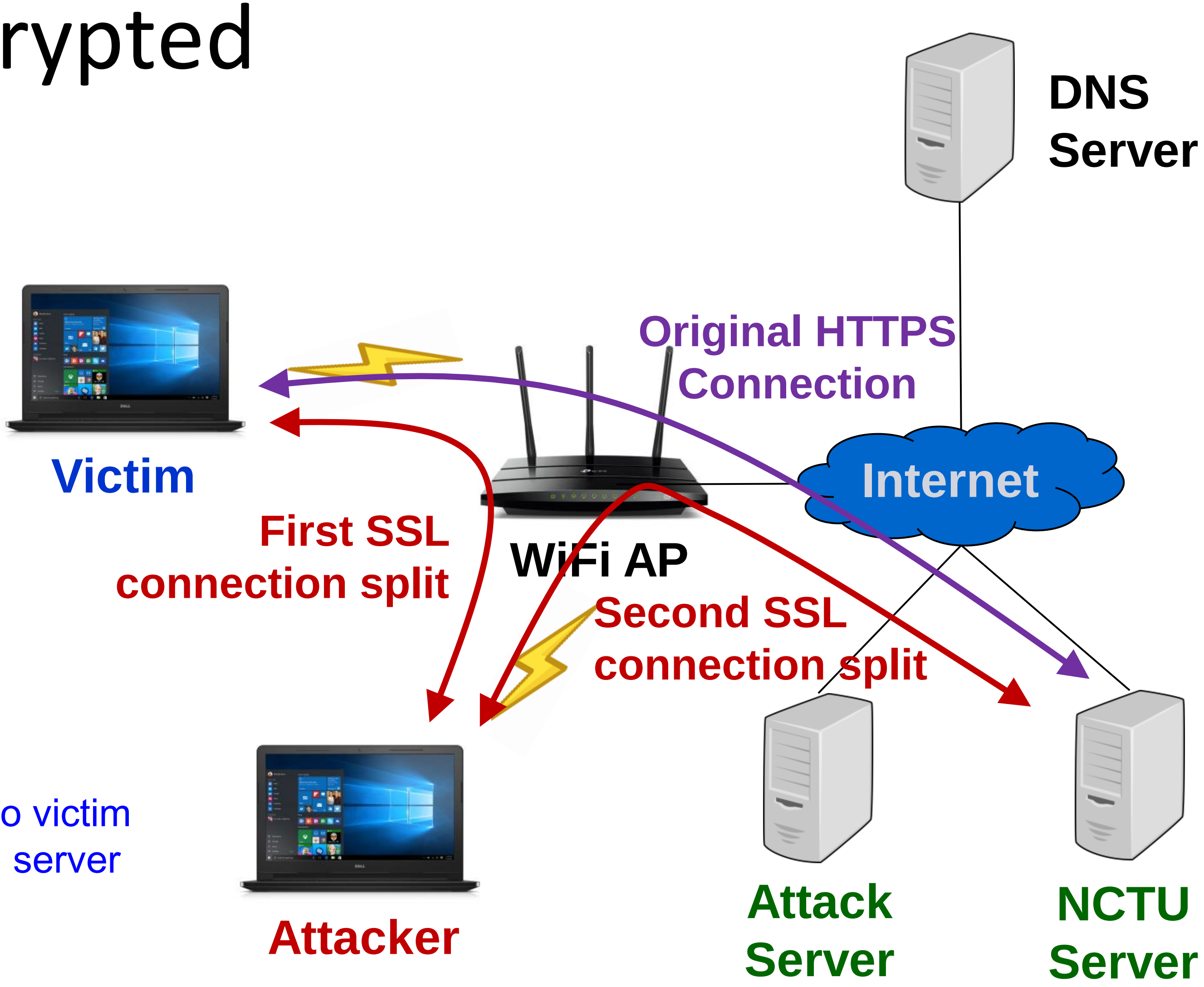
```
No.   Time            Source          Destination     Protocol Length Info
 2743 152.022717290 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=1
 2744 152.025483284 192.168.0.100   8.8.8.8          ICMP     74 Echo (ping) request  id=0x0001, seq=327/18177, ttl=1
 2745 152.049717459 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=1
 2746 152.053411633 8.8.8.8         192.168.0.100    ICMP     74 Echo (ping) reply    id=0x0001, seq=327/18177, ttl=1
▸ Frame 2746: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_45:5c:bb (b8:76:3f:45:5c:bb), Dst: IntelCor_7e:32:81 (00:e1:8c:7e:32:81)
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.0.100
▸ Internet Control Message Protocol
```

Attacker ➔ Victim

Task III: SSL Split on Encrypted SSL/TLS Connections

- Split SSL/TLS connections
  - □ You can use ‘sslsplit’ command as a tool for this attack against encrypted network connection
  - □ You are allowed to install certificates on the victim

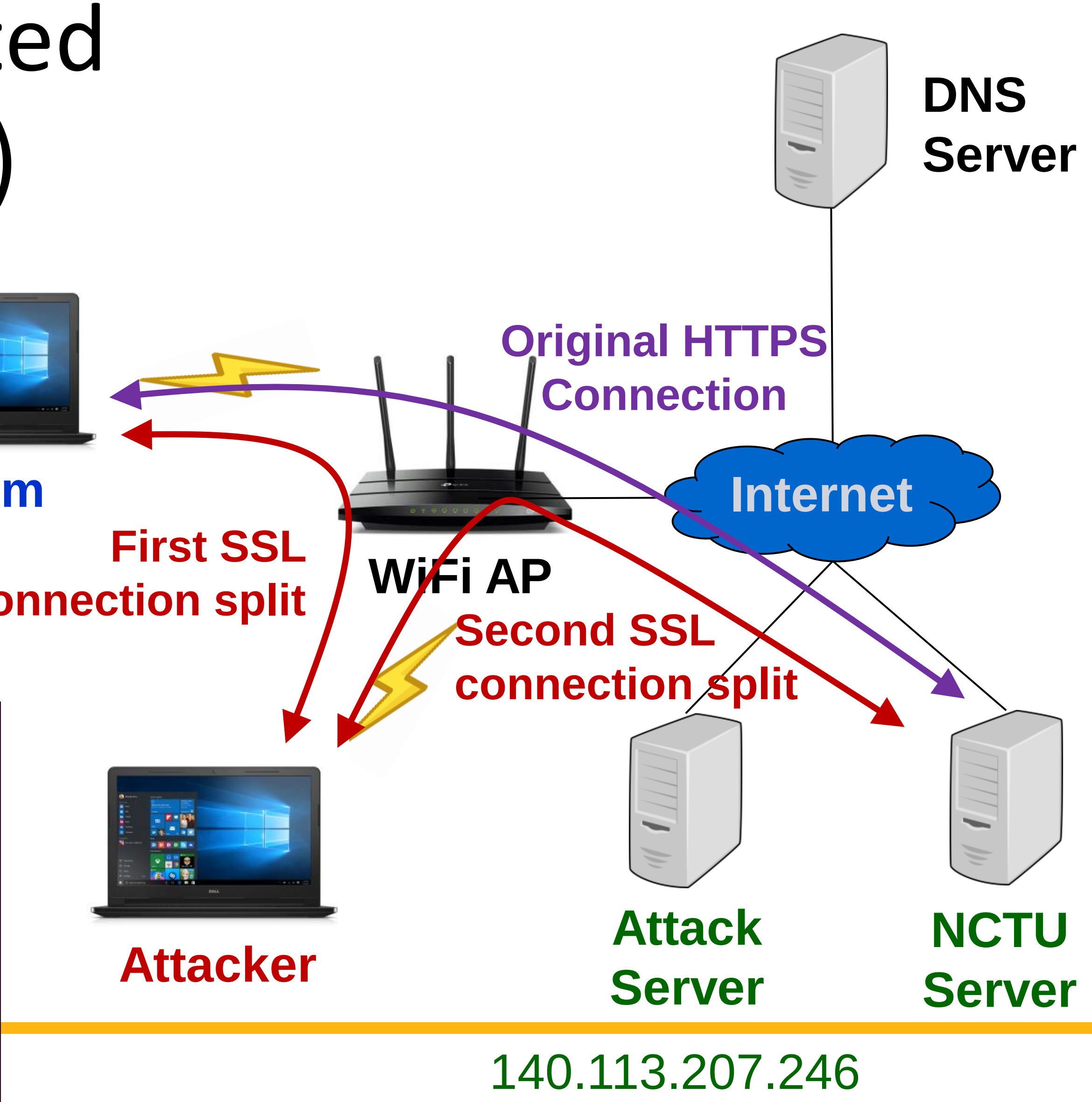
attacker需要傳送certificate to victim
也要傳送certificate to NCTU server

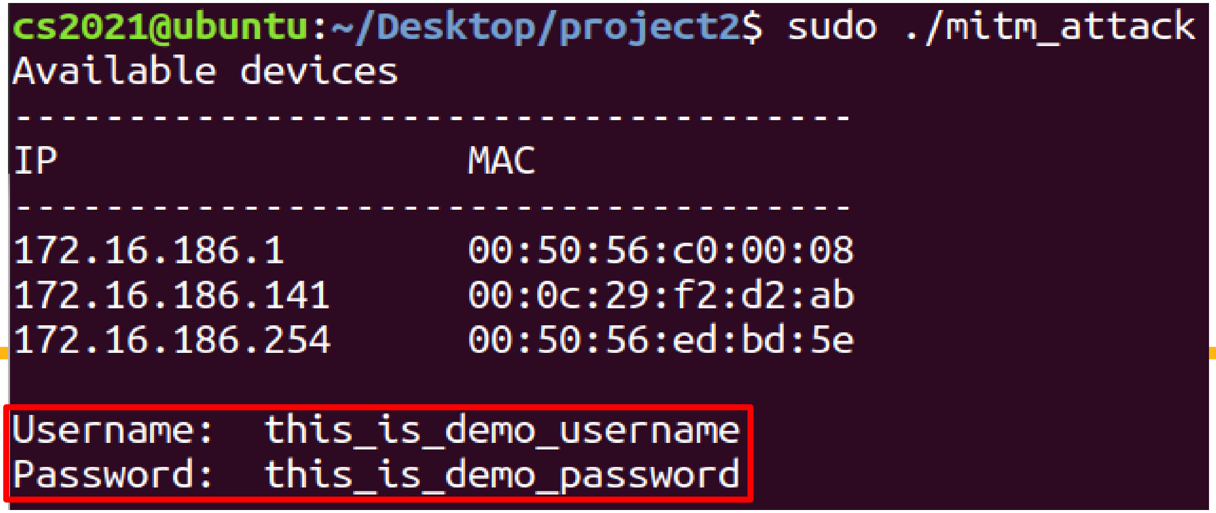

DNS Server
Original HTTPS Connection
Victim
Internet
First SSL connection split
WiFi AP
Second SSL connection split
Attacker
Attack Server
NCTU Server

140.113.207.246

Task III: SSL Split on Encrypted SSL/TLS Connections (Cont.)

- Fetch all the inputted usernames/passwords on a specific web page
    - □ Parse HTTP content to print out usernames/passwords


DNS Server
Original HTTPS Connection
Victim
Internet
First SSL connection split
WiFi AP
Second SSL connection split
Attacker
Attack Server
NCTU Server
140.113.207.246

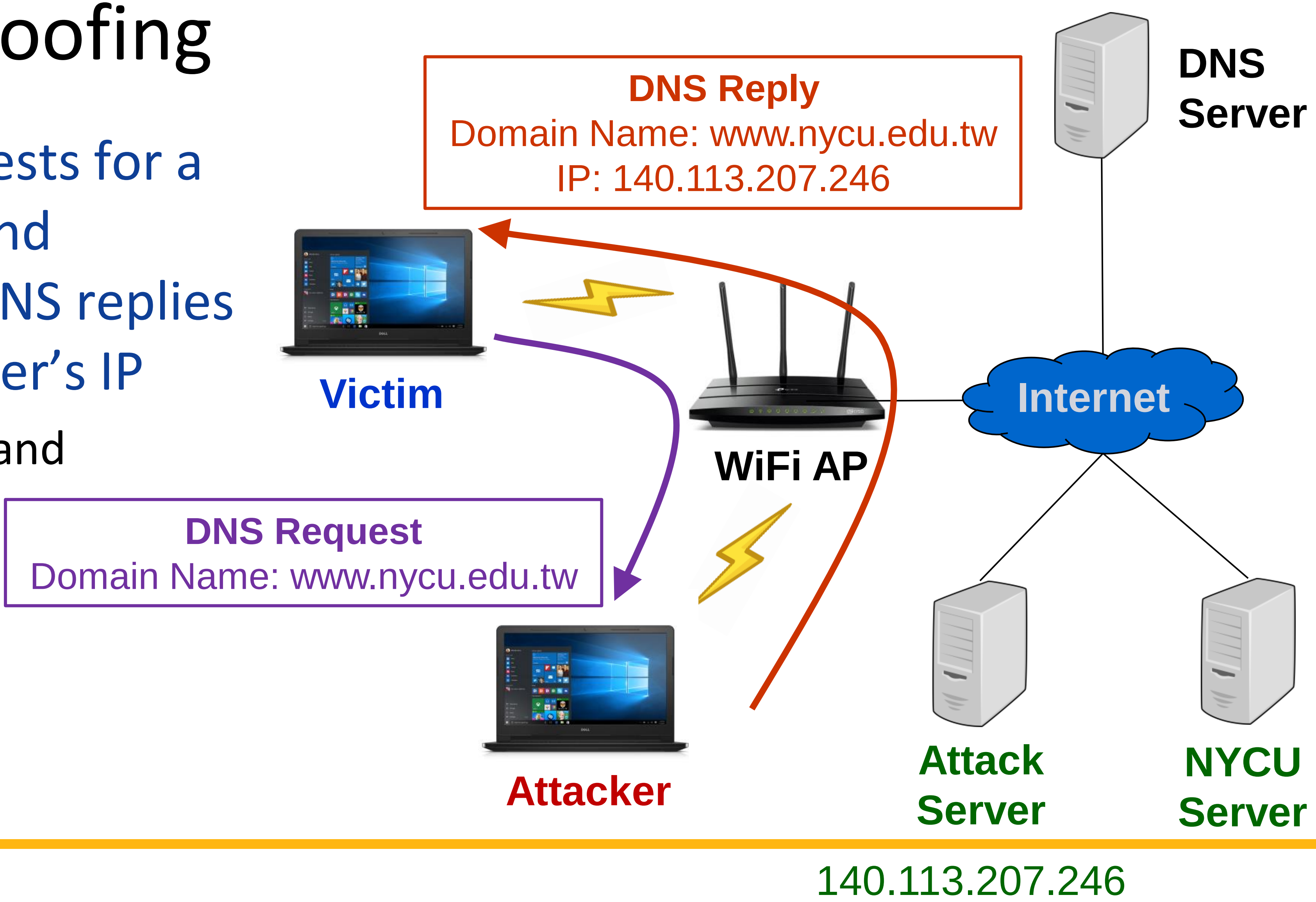


```
cs2021@ubuntu:~/Desktop/project2$ sudo ./mitm_attack
Available devices
---------------------------------------
IP                     MAC
---------------------------------------
172.16.186.1           00:50:56:c0:00:08
172.16.186.141         00:0c:29:f2:d2:ab
172.16.186.254         00:50:56:ed:bd:5e

Username:  this_is_demo_username
Password:  this_is_demo_password
```

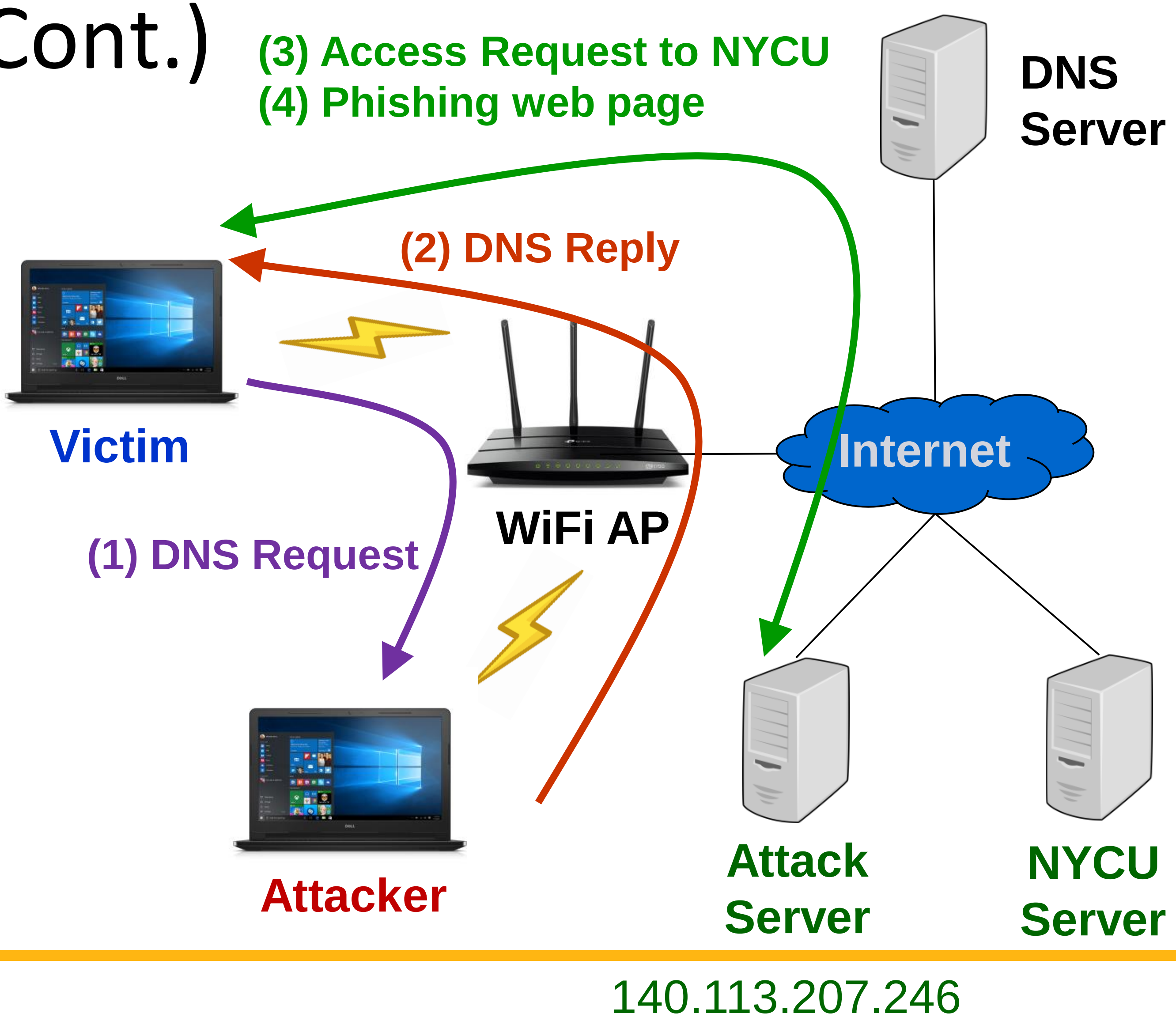
Task IV: DNS Spoofing

- Intercept DNS requests for a specific web page and generate spoofed DNS replies with the attack server's IP
  - □ You can use 'scapy' and 'NetfilterQueue' library in Python



Task IV: DNS Spoofing (Cont.)

- Successful attack
  - □ An access request to NYCU home page will be redirected to the attack server (140.113.207.246)
  - □ A phishing web page will be shown to Victim

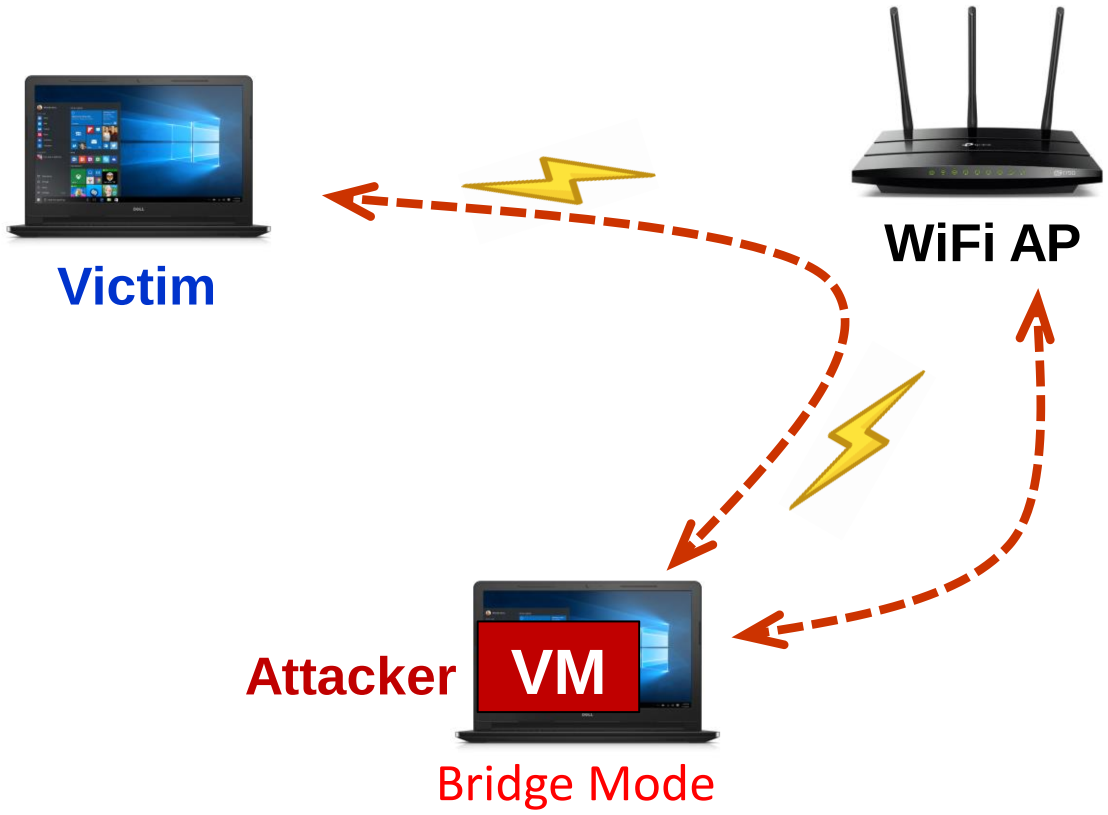


Requirements

- You need to develop/run your program in a given virtual machine
  - VM image: Please download it from Link
    - Username/password: cs2021/cs2021
- You are allowed to use C/C++ and Python
- You are allowed to team up. Each team has at most 2 students
  - Teams: discussions are allowed, but no collaboration
- Please submit your source codes and report to New E3

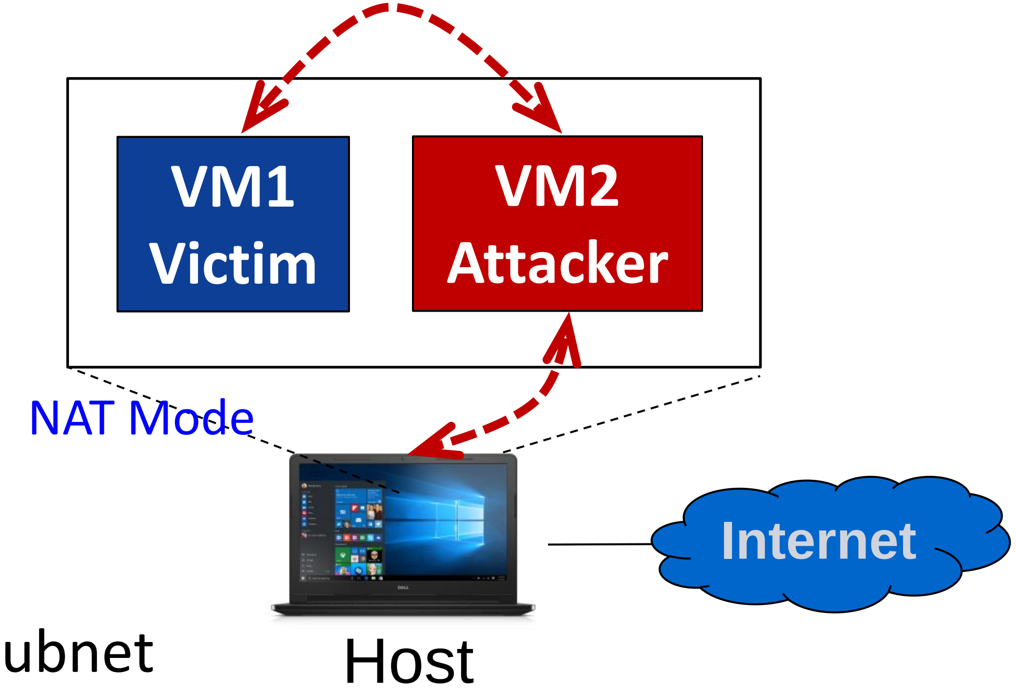
Test Scenario I: Target Scenario

- However, this scenario does not work for all the combinations of OS and VM software
  - □ Working: Linux + VirtualBox/VMware
  - □ Not working properly
    - Windows + VirtualBox/Vmware
    - MacOS + VirtualBox
- You can choose Test Scenario II



Test Scenario II: Alternative Scenario

- VM2 (Attacker) launches attacks on VM1 (Victim)
  - NAT mode shall be used for VMs



- Host is similar to the role of the AP in Test Scenario I
  - Scenario I: The Wi-Fi devices are in the same subnet
  - Scenario II: The VMs are in the same subnet
- Host can be connected to the Internet via Wi-Fi or wired Ethernet

Important: How to Prepare Your Attack Programs?

- Must provide a **Makefile** which compiles your source codes into two executable files, named **mitm_attack** and **pharm_attack** (Missing: -20%)
- Test requirements for the programs
  - Must be run in the given VM without any additional tools or libraries
  - Must use the following parameters
    - Test web page in the man-in-the-middle attack: https://e3.nycu.edu.tw/login/index.php
    - DNS spoofing for the NYCU home page: http://www.nycu.edu.tw
    - Attacker server IP in the DNS spoofing: 140.113.207.246
  - Must work for the test commands: ./mitm_attack and ./pharm_attack

Important: How to Prepare Your Attack Programs?

- Results from the MITM attack (./mitm_attack)
  - □ Print out the IP/MAC addresses of all the Wi-Fi devices or VMs except for Attacker and AP/Host
  - □ Print out the username and password which a user submits to the website https://e3.nycu.edu.tw/login/index.php using any of the Wi-Fi devices or VMs
- Results from the pharming attack (./pharm_attack)
  - □ Print out the IP/MAC addresses of all the Wi-Fi devices or VMs except for Attacker and AP/Host
  - □ Redirect the NYCU home page (www.nycu.edu.tw) to the phishing page (140.113.207.246)
- Demo
  - □ Verify the MITM attack by giving inputs on the website using one Wi-Fi device or VM
  - □ Verify the pharming attack by accessing the NYCU page on one Wi-Fi device or VM

Important: How to Prepare Your Report?

- Item 1 (5%): please give evidence that you have finished the MITM attack
  - □ Specify your scenario (I or II) and Illustrate your results based on some snapshots
- Item 2 (5%): please give evidence that you have finished the pharming attack
  - □ Specify your scenario (I or II) and Illustrate your results based on some snapshots
- Item 3 (10%): please propose a solution that can defend against the ARP spoofing attack
  - □ No more than 200 English words
- Note: the report must be written in English with font size 11 or 12 in Times New Roman. It must be submitted in one PDF file with a name “report.pdf.”

Project Submission

- Due date: 4/23 11:55pm
- Makeup submission (75 points at most): TBA (After the final)
- Submission rules
  - Put all your files into a directory and name it using your student ID(s)
    - If your team has two members, please concatenate your IDs separated by "-"
  - Zip the directory and upload the zip file to New E3
  - A sample of the zip file: 01212112-02121221.zip
    - Makefile
    - mitm_attack.cpp
    - report.pdf
    - mitm_attack.h
    - ….

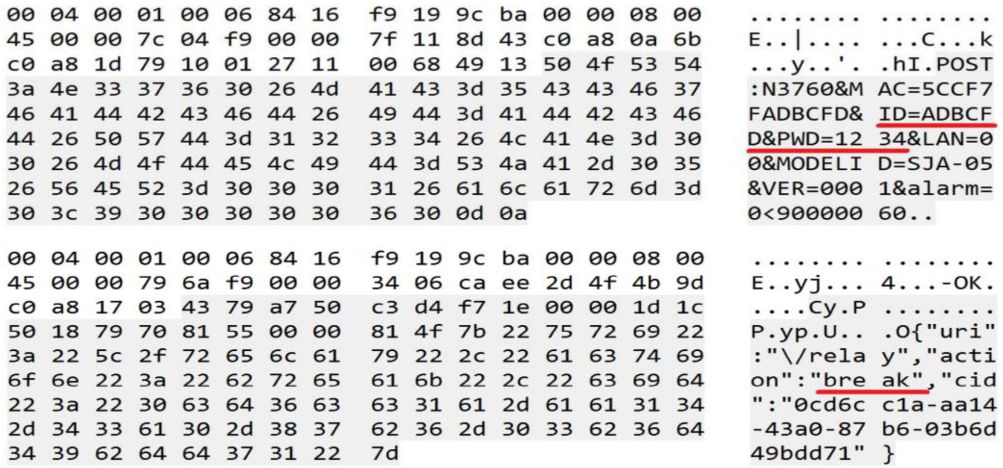
Project Demo

- Date: TBA
- Makeup submission (75 points at most): TBA (After the final)
- TA will prepare your zip file for you to demo
- You will
  - be asked to reproduce your MITM and pharming attacks
  - be only allowed to “make” to compile all your files, and run your attack binary programs or scripts
  - be not allowed to modify your codes or scripts
  - be asked some questions
    - E.g., How did you implement the SSL split? How did you resolve the certificate issue?
  - be responsible to show the outcome to TA and explain why you have successfully achieved the goals

Bonus: Finding Insecure Wi-Fi IoT Devices (TBA)

- Some IoT devices have no security protocol support or no resource to check the servers' certificates or
- Final score +2.5 points for each identified vulnerable IoT device (at most two)
- Deadline: at the end of May (TBA)

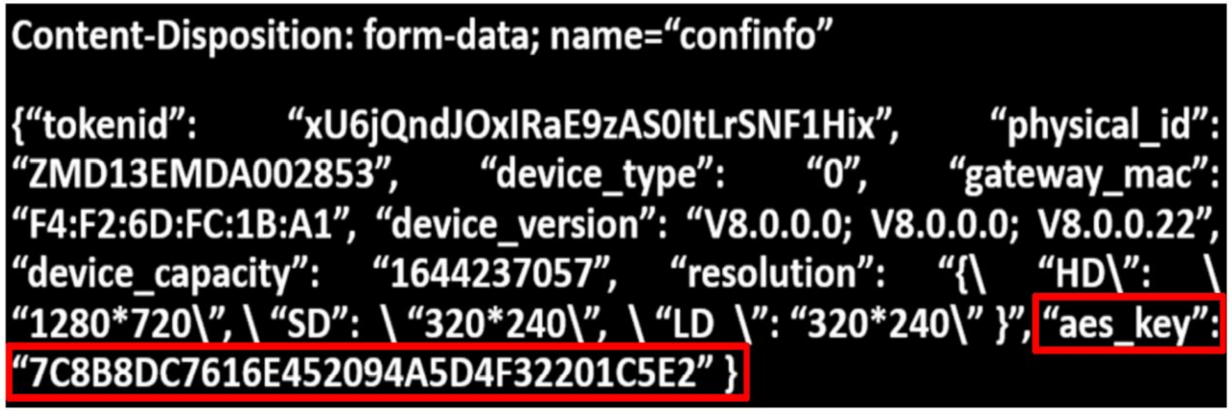
Smart plug: no security support → Hijacking attack



```
00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 7c 04 f9 00 00  7f 11 8d 43 c0 a8 0a 6b   E..|.... ...C...k
c0 a8 1d 79 10 01 27 11  00 68 49 13 50 4f 53 54   ...y..'. .hI.POST
3a 4e 33 37 36 30 26 4d  41 43 3d 35 43 43 46 37   :N3760&M AC=5CCF7
46 41 44 42 43 46 44 26  49 44 3d 41 44 42 43 46   FADBCFD& ID=ADBCF
44 26 50 57 44 3d 31 32  33 34 26 4c 41 4e 3d 30   D&PWD=12 34&LAN=0
30 26 4d 4f 44 45 4c 49  44 3d 53 4a 41 2d 30 35   0&MODELI D=SJA-05
26 56 45 52 3d 30 30 30  31 26 61 6c 61 72 6d 3d   &VER=000 1&alarm=
30 3c 39 30 30 30 30 30  36 30 0d 0a               0<900000 60..

00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 79 6a f9 00 00  34 06 ca ee 2d 4f 4b 9d   E..yj... 4...-OK.
c0 a8 17 03 43 79 a7 50  c3 d4 f7 1e 00 00 1d 1c   ....Cy.P ........
50 18 79 70 81 55 00 00  81 4f 7b 22 75 72 69 22   P.yp.U.. .O{"uri"
3a 22 5c 2f 72 65 6c 61  79 22 2c 22 61 63 74 69   :"\/rela y","acti
6f 6e 22 3a 22 62 72 65  61 6b 22 2c 22 63 69 64   on":"bre ak","cid
22 3a 22 30 63 64 36 63  63 31 61 2d 61 61 31 34   ":"0cd6c c1a-aa14
2d 34 33 61 30 2d 38 37  62 36 2d 30 33 62 36 64   -43a0-87 b6-03b6d
34 39 62 64 64 37 31 22  7d                        49bdd71" }
```

Camera: no certificate check → Spying attack



```
Content-Disposition: form-data; name="confinfo"

{"tokenid": "xU6jQndJOxIRaE9zAS0ItLrSNF1Hix", "physical_id": "ZMD13EMDA002853", "device_type": "0", "gateway_mac": "F4:F2:6D:FC:1B:A1", "device_version": "V8.0.0.0; V8.0.0.0; V8.0.0.22", "device_capacity": "1644237057", "resolution": "{\ "HD\": \ "1280*720\", \ "SD": \ "320*240\", \ "LD \": "320*240\" }", "aes_key": "7C8B8DC7616E452094A5D4F32201C5E2" }
```

Reference: Lei et al. "SecWIR: Securing Smart Home IoT Communications via Wi-Fi Routers with Embedded Intelligence," ACM Mobisys 2020.

Questions?