Computer Security Capstone

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

Chi-Yu Li (2024 Spring)
Computer Science Department
National Yang Ming Chiao Tung University

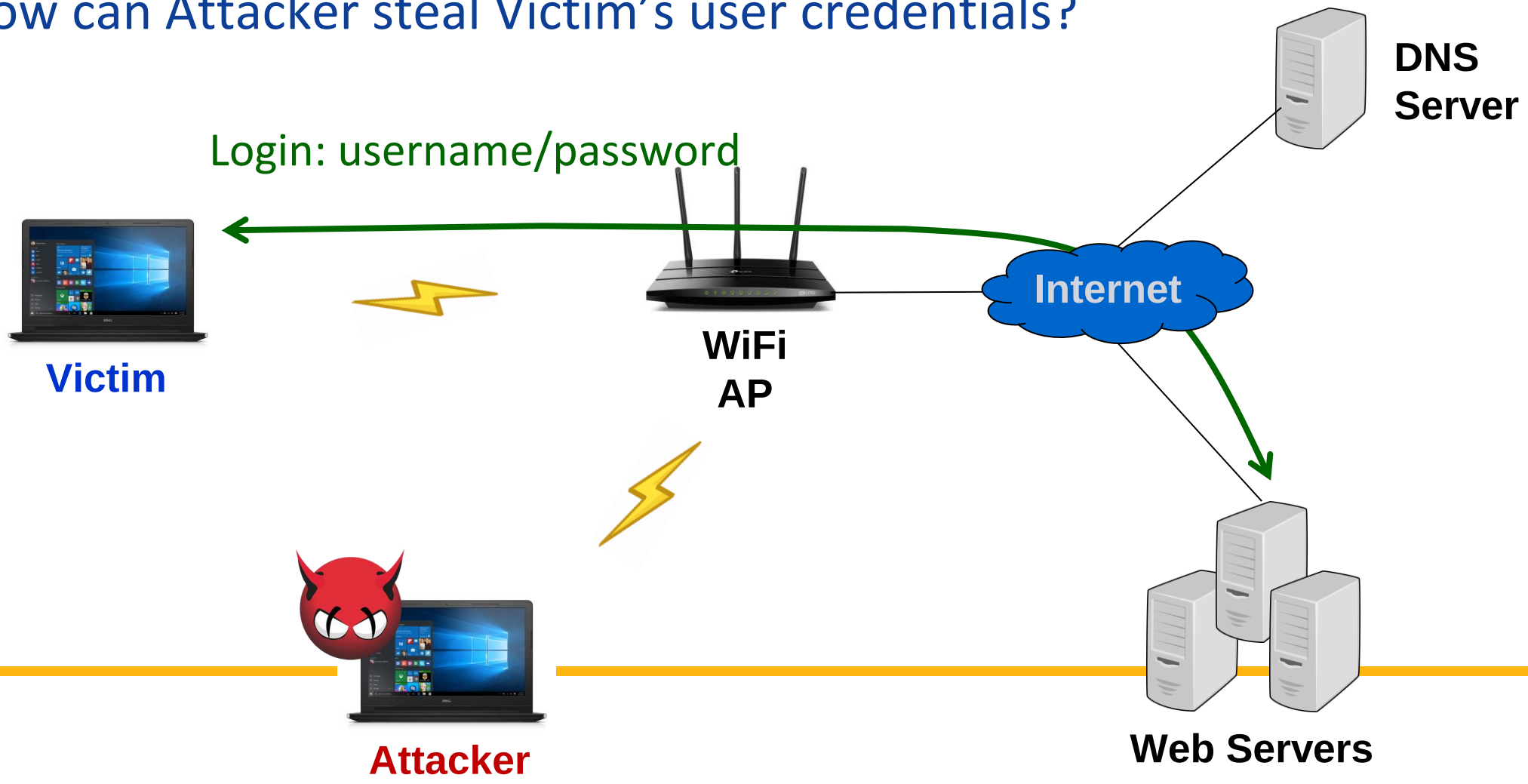
Goal

- 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

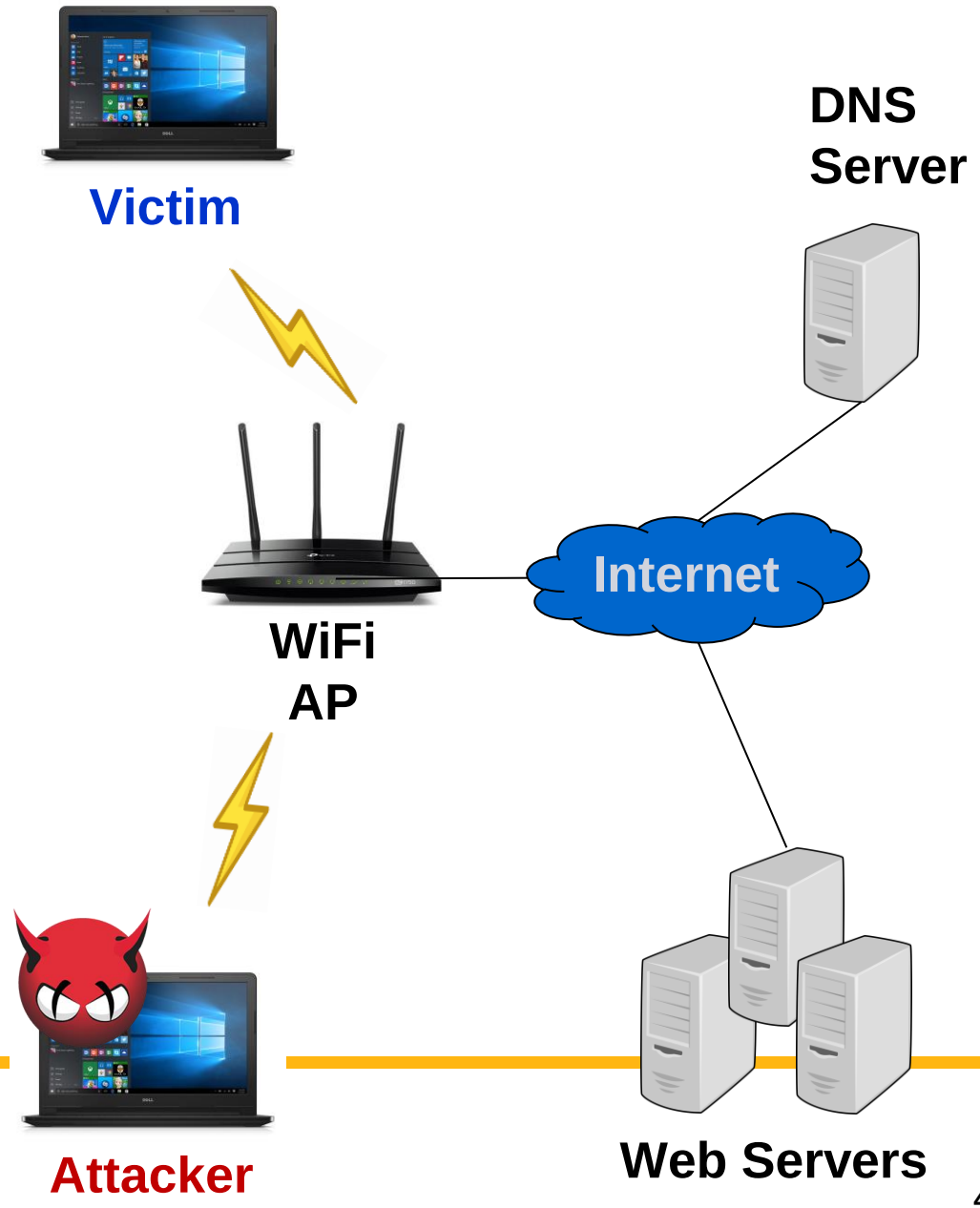
Attack Scenario

- How can Attacker steal Victim's user credentials?



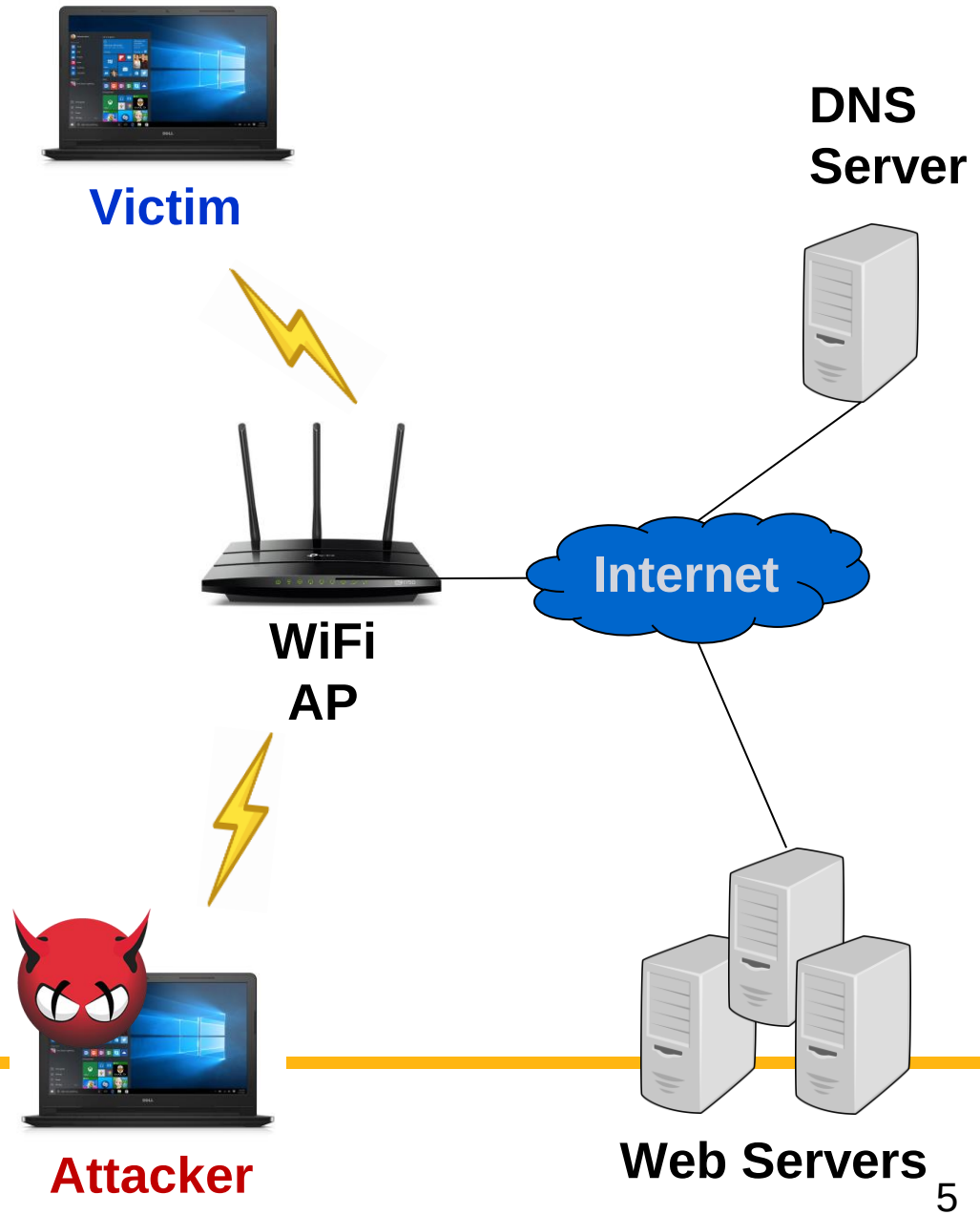
Major Ideas

- Redirect Victim's traffic to Attacker
  - Man-in-the-middle based on ARP spoofing
  - How to know Victim's IP/MAC address?
- How about HTTP sessions?
  - MITM attack: Parse the packet to obtain user credentials
  - Pharming attack: redirect HTTP requests to a phishing web page



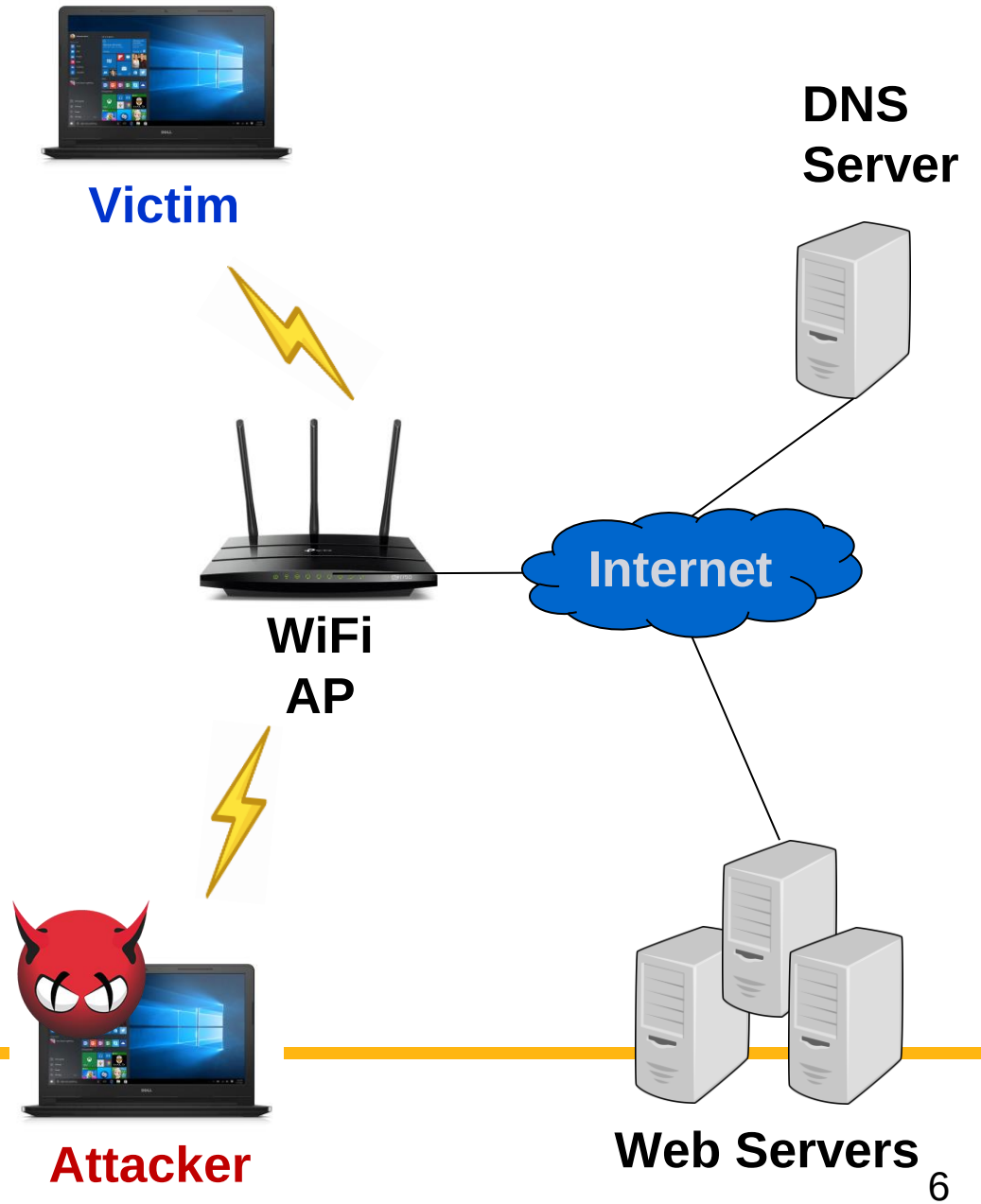
Tasks: MITM and Pharming

- MITM Attack (60%)
  - 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%)
  - Fetch the inputted username/password strings from HTTP sessions (Task III: 15%)
  - One implementation question during the demo (10%)



Tasks: MITM and Pharming

- Pharming Attack (40%)
  - Obtain all other client devices' IP/MAC addresses in a connected Wi-Fi network
  - DNS spoofing attack for web services (Task IV: 30%)
  - One implementation question during the demo (10%)



Task I: Device Address Information Collection

- Scan all the devices' IP/MAC addresses in the Wi-Fi network
  - Hint: ARP format and raw socket.

```
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
```

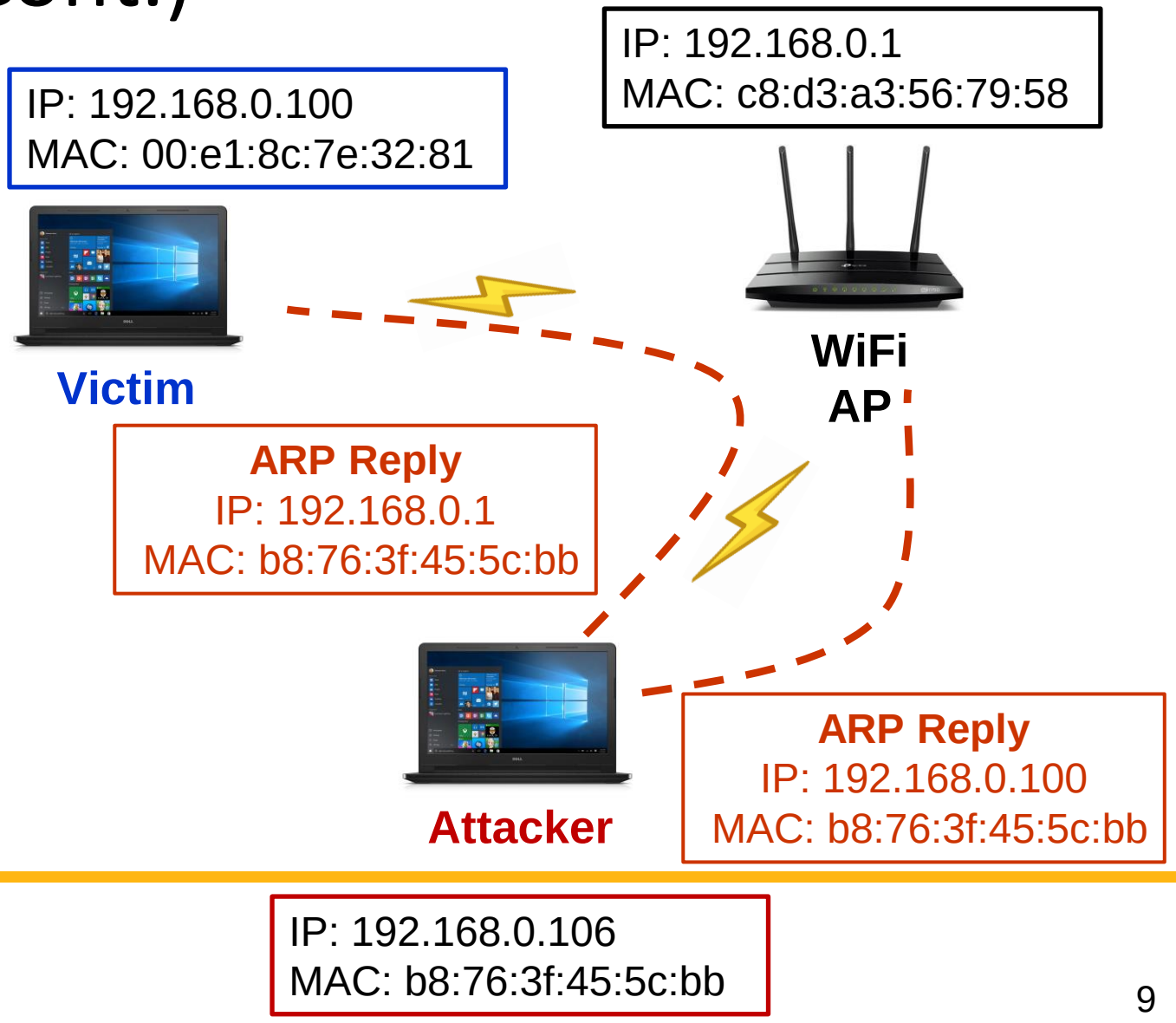
- Fetch the IP/MAC addresses of all the other client devices

Task II: ARP Spoofing

- What is ARP (Address Resolution 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

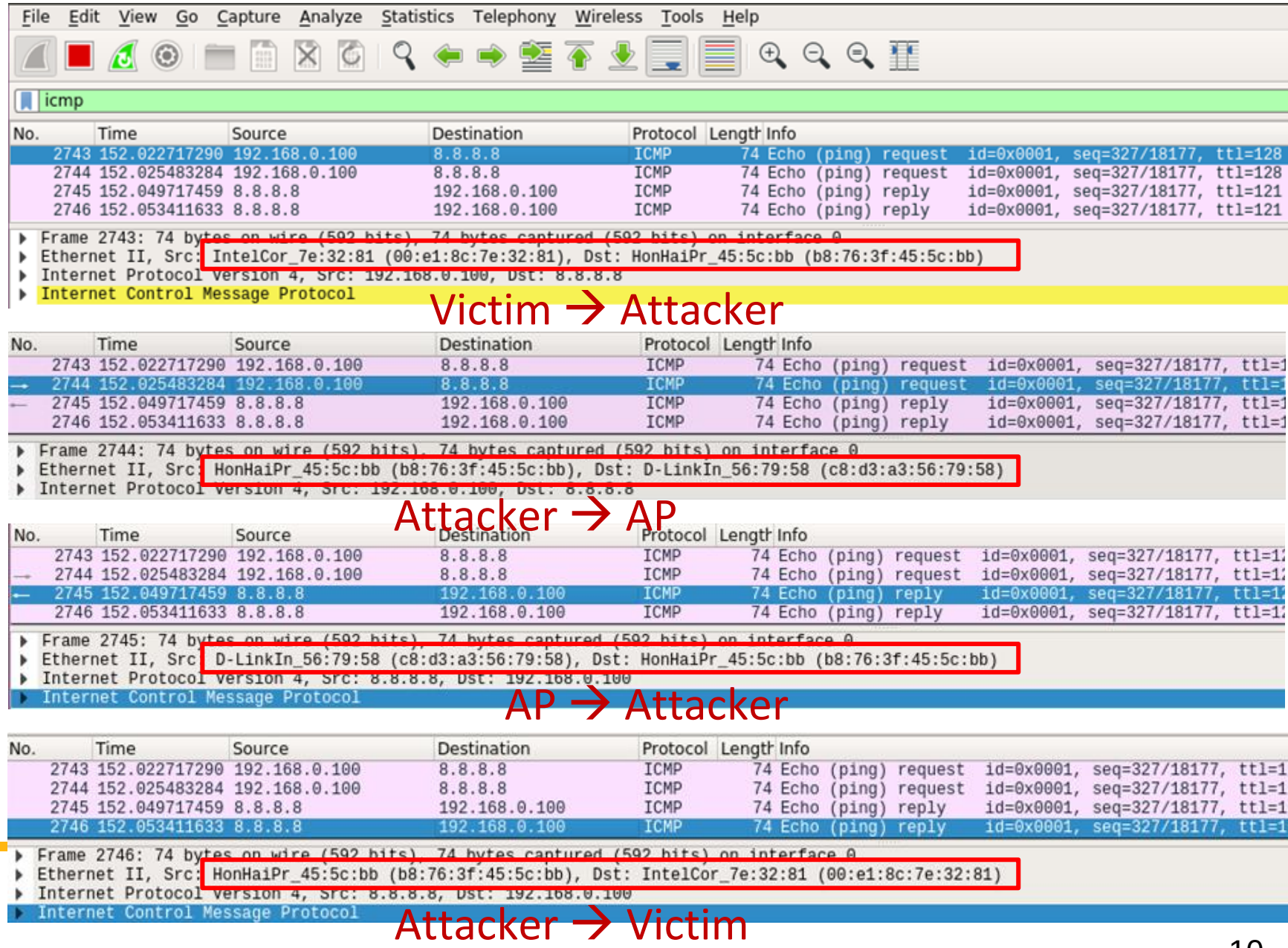
Task II: ARP Spoofing (Cont.)

- Generate spoofed ARP replies for all other client devices
  - Hint: ARP format and thread
- Both uplink and downlink should be considered
  - Other client devices' network services can work normally



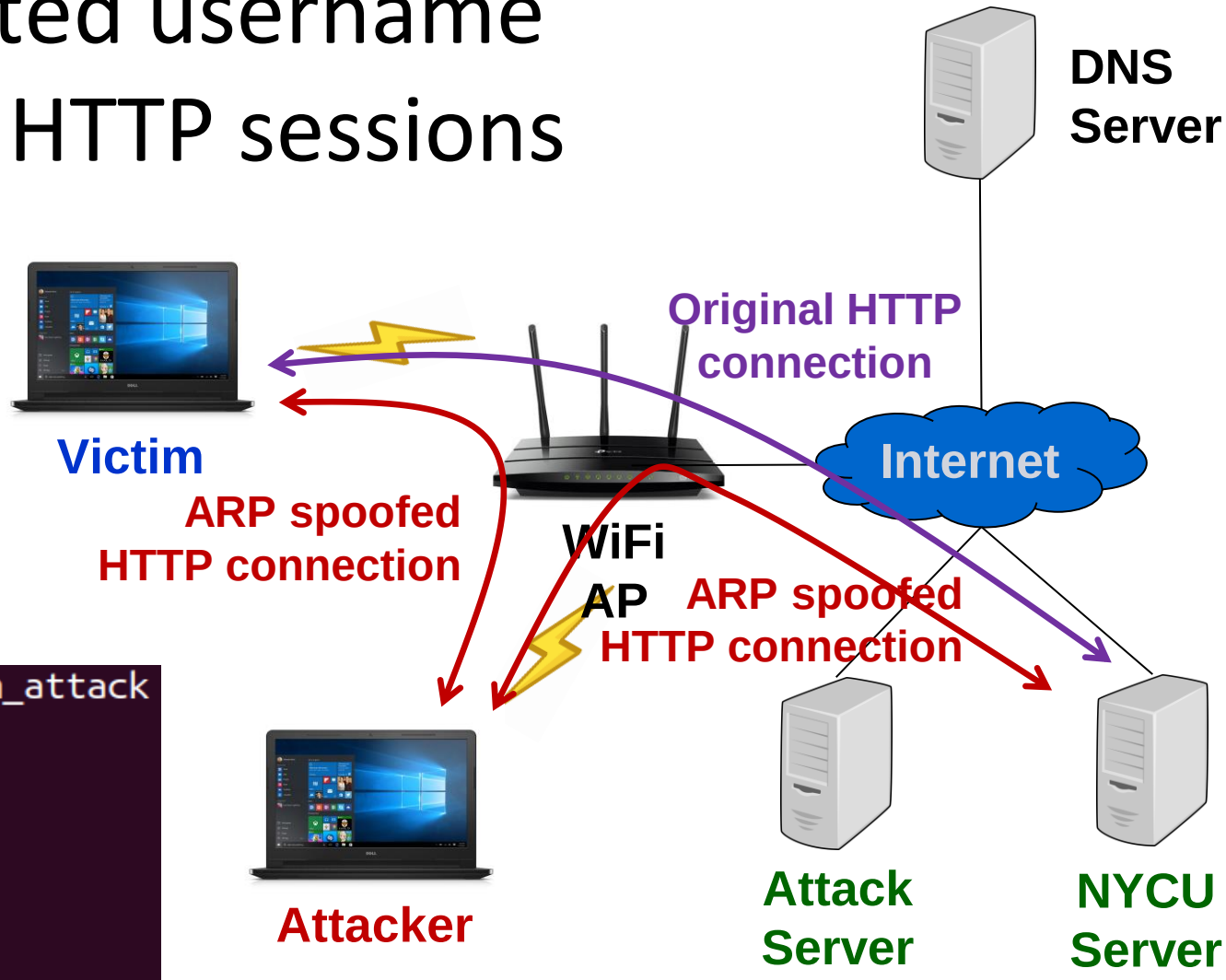
Task II: ARP Spoofing (Cont.)

- An example trace of the successful ARP spoofing at Attacker



Task III: Fetch the inputted username /password strings from HTTP sessions

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

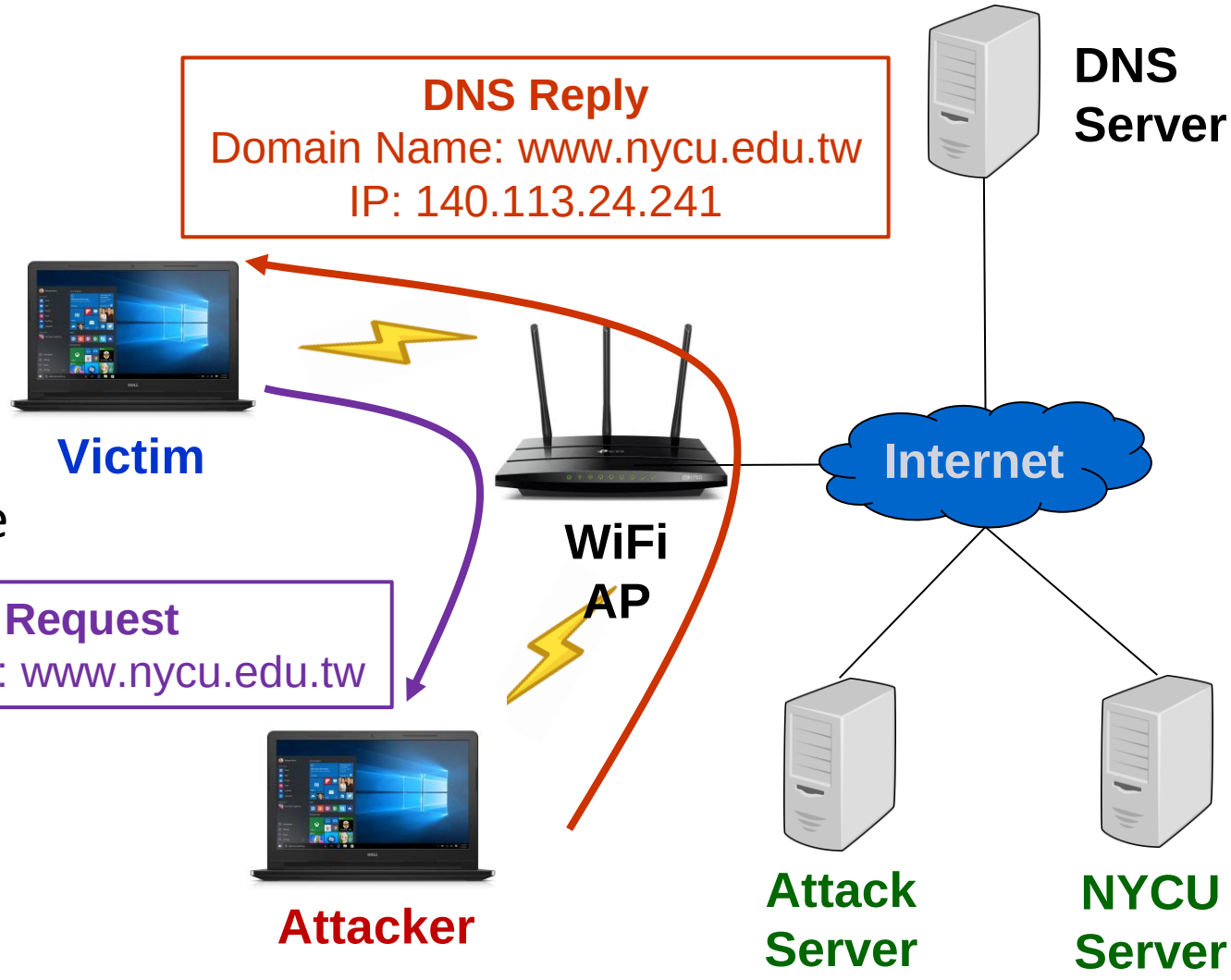



```
cs2021@ubuntu:~/Desktop/project2$ sudo ./mitm_attack
Available devices
-------------------------------------
IP                  MAC
-------------------------------------
172.16.186.1        00:50:56:c0:00:08
172.16.186.141      00:0c:29:f2:d2:ab
172.16.186.254      00:50:56:ed:bd:5e

Username:  this_is_demo_username
Password:  this_is_demo_password
```

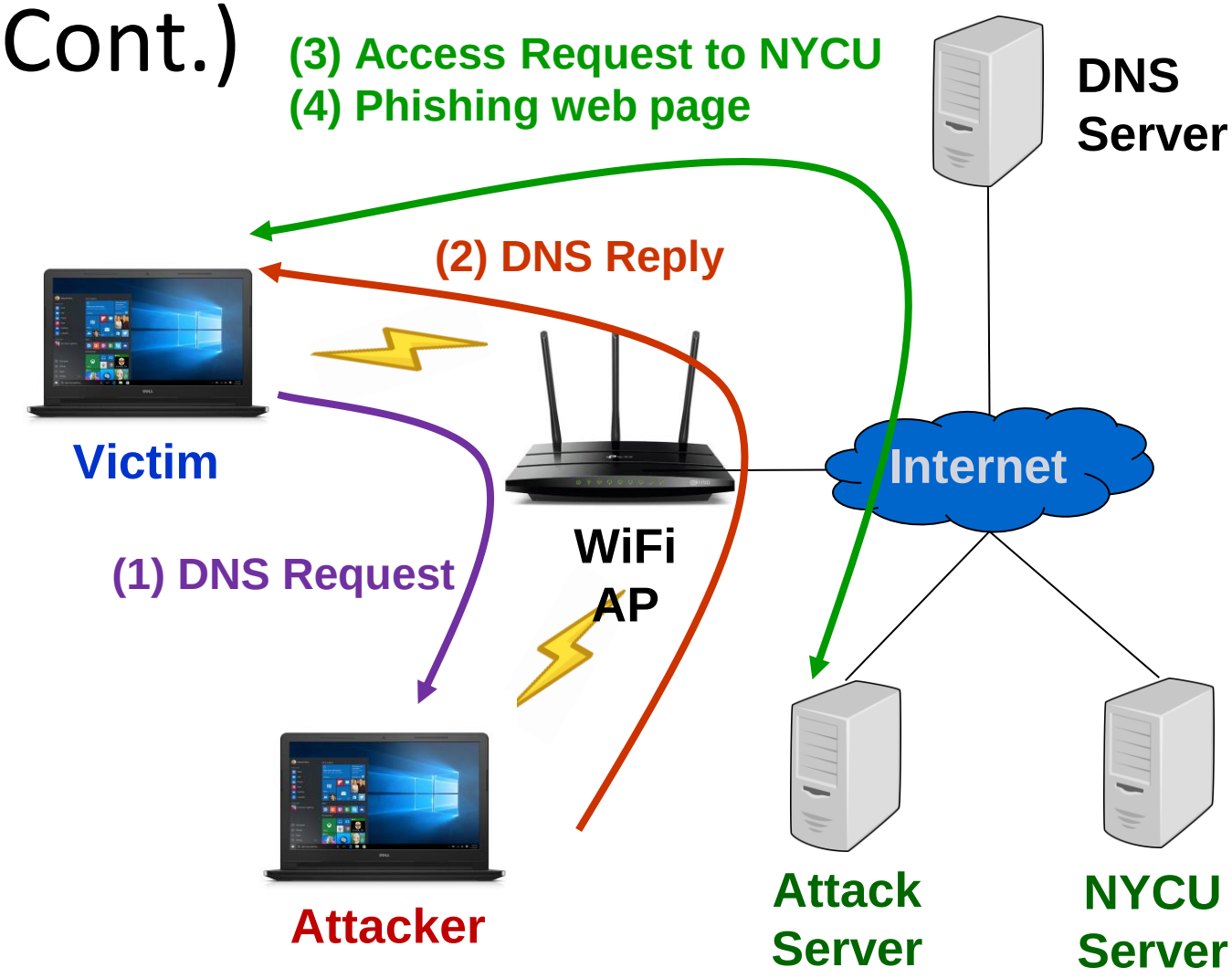
Task IV: DNS Spoofing

- Intercept DNS requests for a specific web page and generate spoofed DNS replies with the attack server's IP
  - Hint: DNS format, Netfilter queue



Task IV: DNS Spoofing (Cont.)

- Successful attack
  - An access request to NYCU home page will be redirected to the attack server (140.113.24.241)
  - A phishing web page will be shown to Victim

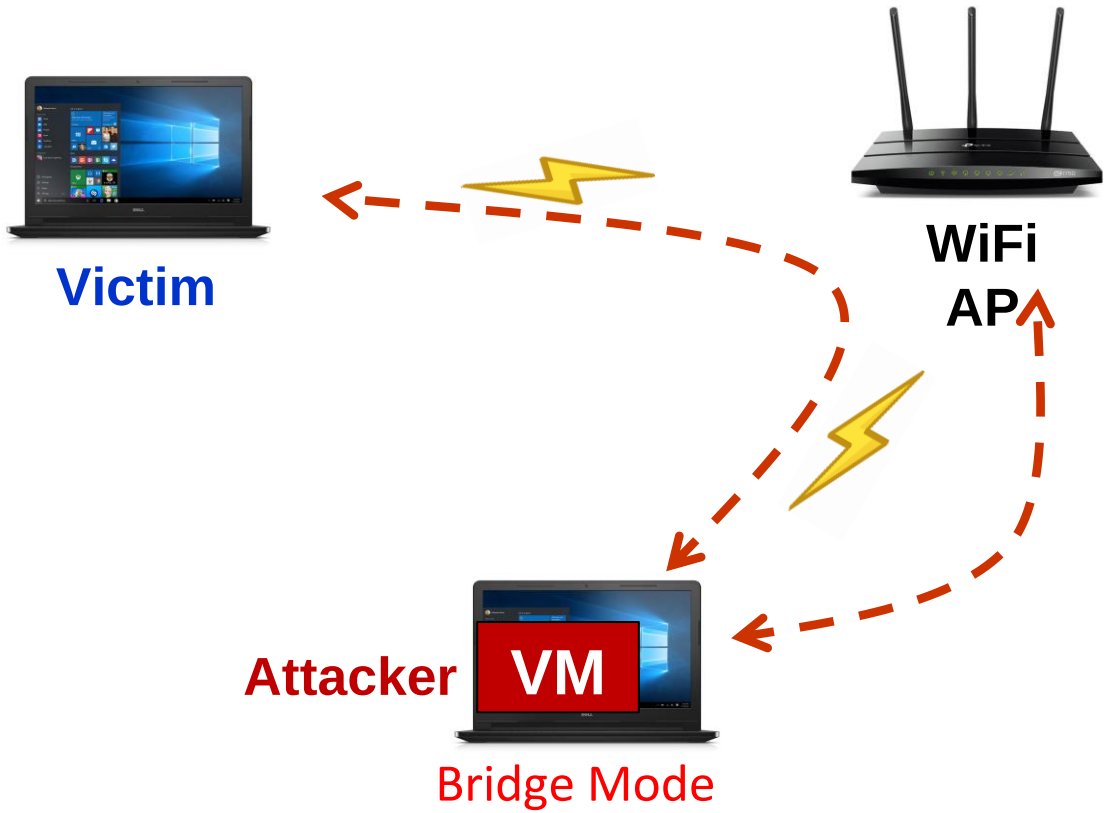


Requirements

- You need to develop/run your program in a given virtual machine
  - VM image: Please download it from Link
    - Username/password: csc2024/csc2024
- Only C/C++ is allowed for the development
  - To be better familiar with the protocols (Python is not allowed)
- 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 to New E3

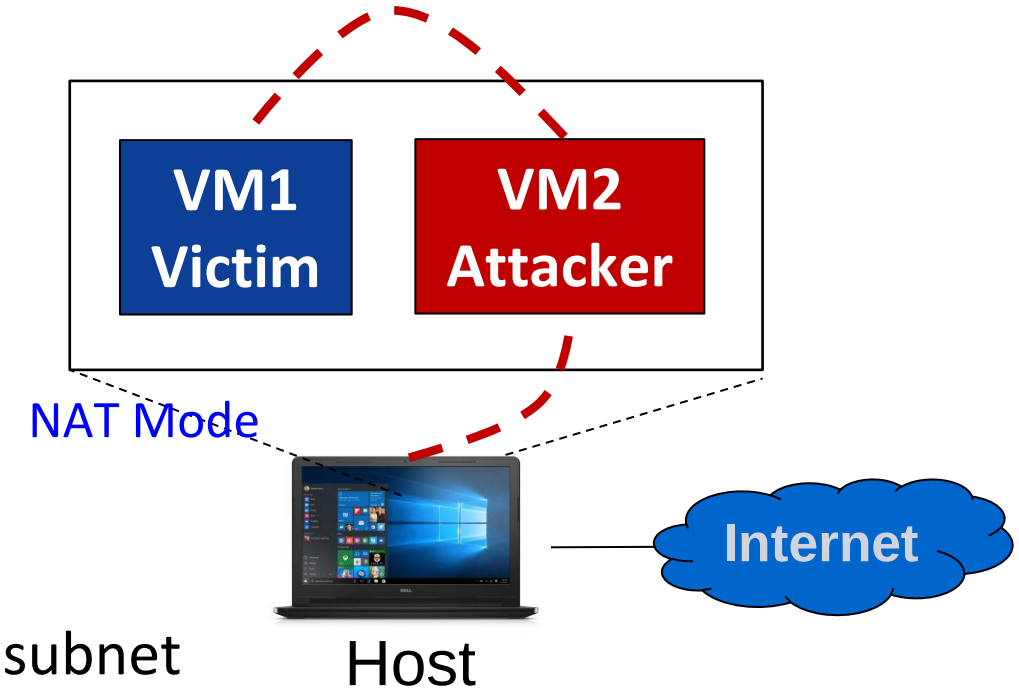
Test Scenario I: Target Scenario

- 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: http://vbsca.ca/login/login.asp (use private window)
    - DNS spoofing for the NYCU home page: http://www.nycu.edu.tw (use private window)
    - Attacker server IP in the DNS spoofing: 140.113.24.241
  - 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 http://vbsca.ca/login/login.asp 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 (140.113.24.241)
- Demo
  - Verify the MITM attack by giving inputs on the website using one Wi-Fi device or VM
    - Only allowed to run ./mitm_attack (No manual configuration is allowed)
  - Verify the pharming attack by accessing the NYCU page on one Wi-Fi device or VM
    - Only allowed to run ./pharm_attack (No manual configuration is allowed)

Project Submission

- Due date: 4/24 11:55pm
  - Late submission is not allowed
- Makeup submission (75 points at most): TBA (After the final)
- Submission rules (Wrong file name or format will result in 10 points deduction)
  - 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
    - mitm_attack.h
    - ….

Online Project Demo

- Date: 4/26
- Makeup submission (75 points at most): TBA (After the final)
- TA will prepare your zip file and run your programs for the demo on behalf of you
- 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
  - be responsible to show and explain the outcome to TA

Hint 1: ARP Spoofing

- ARP format

Hardware Type (16 bits)		Protocol Type (16 bits)
Hardware Length (4 bits)	Protocol Length (4 bits)	Operation (16 bits)
Sender Hardware Address (32 bits)		
Sender Protocol Address (32 bits)		
Target Hardware Address (32 bits)		
Target Protocol Address (32 bits)		

Hint 2: DNS Spoofing

- DNS format

Identification (16 bits)	Flags (16 bits)
Number of questions (16 bits)	Number of answer RRs (16 bits)
Number of authority RRs (16 bits)	Number of additional RRs (16 bits)
Questions (variable bits)	
Answers (variable bits)	
…	

Header
12 bytes

Hint 2: DNS Spoofing (Cont.)

- DNS packets may employ message compression
  - Reference: https://datatracker.ietf.org/doc/html/rfc1035#section-4.1.4

Hint 3: Netfilter queue

- Functions:
  - **nfq_open():** open a nfqueue handler.
  - **nfq_close():** close a nfqueue handler.
  - **nfq_unbind_pf():** unbind nfqueue handler from a protocol family.
  - **nfq_bind_pf()**: bind a nfqueue handler to a given protocol family.
  - **nfq_create_queue():**create a new queue handle and return it.
  - **nfq_destroy_queue():** destroy a queue handle.
  - **nfq_set_mode():** set the amount of packet data that nfqueue copies to userspace.
  - **nfq_fd():** get the file descriptor associated with the nfqueue handler.
  - **nfq_handle_packet():** handle a packet received from the nfqueue subsystem.
  - **nfq_get_msg_packet_hdr():** return the metaheader that wraps the packet.
  - **nfq_get_payload():** get payload.
  - **nfq_set_verdict():** issue a verdict on a packet.
- Reference: https://netfilter.org/projects/libnetfilter_queue/doxygen/html/index.html

Hint 3: Netfilter queue(Cont.)

- nfq_set_verdict(struct nfq_q_handle *qh, uint32_t id, uint32_t verdict, uint32_t data_len, const unsigned char *buf):
  - **Only read the packet:** data_len=0, *buf = nullptr
  - **Modify packet:** data_len=new packet length, *buf = Head of buffer that stored new packet data

Questions?