2. Reconnaissance

Jin Hong

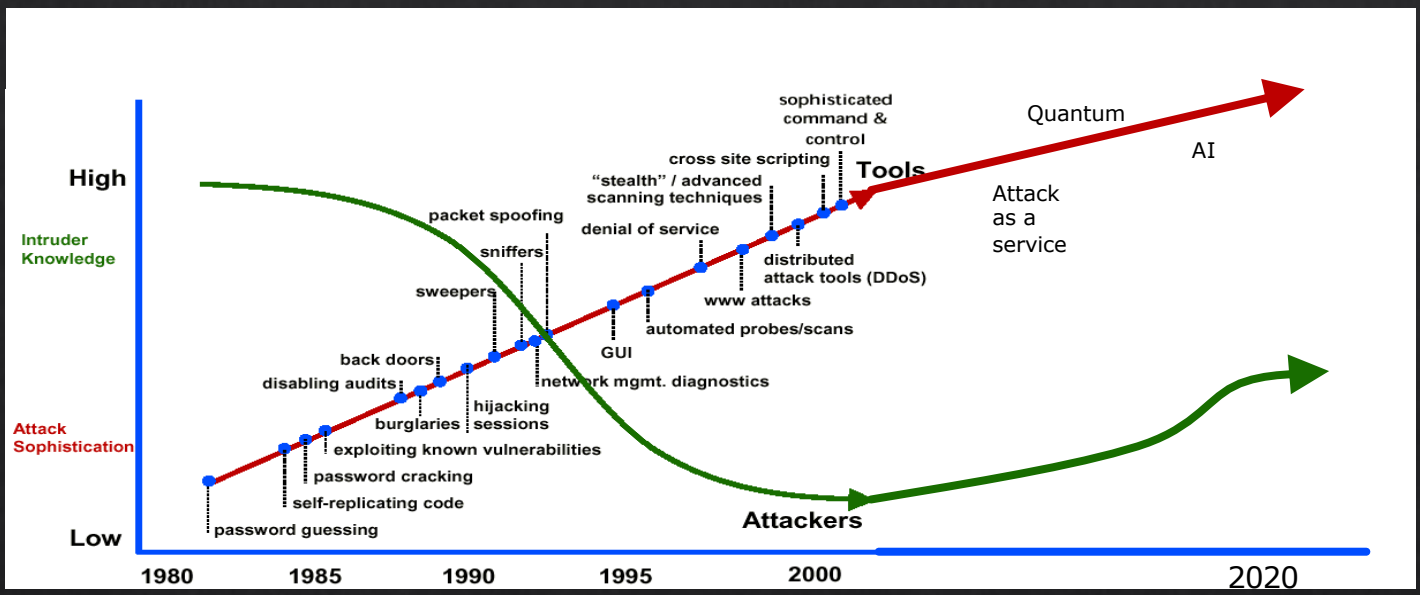
jin.hong@uwa.edu.au

How did it all start…

- Before the Internet, the only way to conduct "cyberattack" is via physical access
  - But the computational power at the time was lacking, did not store much things to steal
- TCP/IP was designed in early 1980s
  - IPv4
- Today, TCP/IP is used everywhere
  - LAN, MAN, WAN, etc
  - Various applications (voice, multimedia etc)

- There are many events that contribute toward attack trends

Attack Trends



Howard Lipson. Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues. CERT Coordination Center. Nov. 2002

Attack Trends

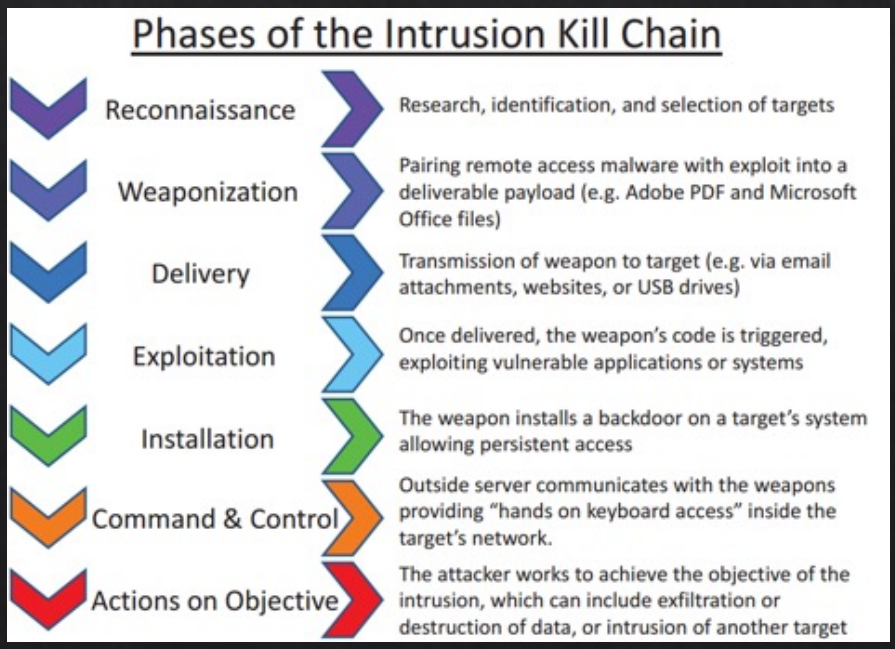
- Attacks are evolving with time

Attack Trends

- What issues do cyberattacks bring?
- Why do people carry out cyberattacks?

Cyber Kill Chain

- Before we talk about any exploits, we must first understand how attacks (usually) happen and progress.
- There are many approaches, but one of the most widely adopted approaches is Cyber Kill Chain
- It was developed by Lockheed Martin (Company) in 2011, specifying 7 high level goals.
- Naturally, breaking the kill chain and its methods became defense against such attacks.



MITRE ATT&CK

- MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.
- This is the current industry standard and most used framework for understanding and communicating how attacks work.
- It goes a step further than the Cyber Kill Chain by expanding the attackers' high-level goals to 14 different tactics.

CKC & MITRE ATT&CK

- Both frameworks capture how attacks progress.
- Lots of common approaches in the framework.
- We will start from the beginning - Reconnaissance

Reconnaissance

"Reconnaissance consists of techniques that involve adversaries actively or passively gathering information that can be used to support targeting. Such information may include details of the victim organization, infrastructure, or staff/personnel."

- MITRE Corp

- Active scanning
  - Scan IP
  - Scan Vulnerability
  - Crawling
- Gather Target Information
  - Hardware
  - Software
  - Firmware
  - Configurations
- Gather Target Identity Information
  - Credentials
  - Emails
  - Names
- Gather Network Information
  - Domain
  - DNS
  - Trust dependencies
  - Topology
  - IP addresses
  - Security appliances
- Gather organization information
  - Location
  - Business relationships
  - Business tempo
  - Roles
- Phishing
  - Service
  - Attachments
  - Links

Etc.

See more details at https://attack.mitre.org/tactics/TA0043/

Reconnaissance

- The very first thing we will try is to discover network hosts.
- There are two main methods of doing this: Passive vs Active
- Passive attacks do not inject any activities on the network
  - E.g., Sniffing
- Active attacks interact with network operations to gather information
  - E.g., Spoofing

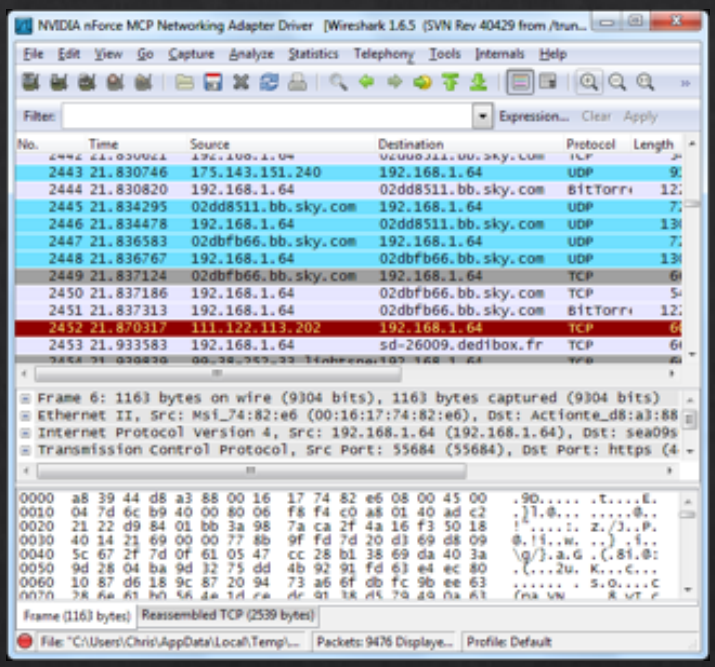
- We will first describe the spoofing materials
- Then we will look at IP scanning and Port scanning (configuration)
  - We will write the code to do this.
  - Don’t worry, you are provided with the code.

Sniffing

- Sniffing is a **passive** attack
- Captures network traffic between two communicating nodes
- Cannot detect such attacks, as the attack does not leave any traces in the system
- Can be mitigated easily using **encryption**
  - But not all things cannot be encrypted in the network – what are they?
- However, establishing a **secure channel** to communicate is important
- As well as the coverage of the channel

Sniffing

- There are a list of **tools** for sniffing
  - E.g., Wireshark, tcpdump, Windows Message Analyzer etc.
- Chrome will notify users that 'http' is **no longer** secure*




```
anuj@packetflows:~$ sudo tcpdump -i eth0
[sudo] password for anuj:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
23:14:09.691884 IP packetflows.local.47860 > productsearch.ubuntu.com.https: Flags
[S], seq 2046377878, win 29200, options [mss 1460,sackOK,TS val 9320277 ecr 0,nop,w
scale 7], length 0
23:14:09.693521 IP packetflows.local.10745 > cdns01.comcast.net.domain: 47145+ PTR?
 49.33.213.162.in-addr.arpa. (44)
23:14:09.693689 IP packetflows.local.10745 > cdns02.comcast.net.domain: 47145+ PTR?
 49.33.213.162.in-addr.arpa. (44)
23:14:09.727692 IP cdns01.comcast.net.domain > packetflows.local.10745: 47145 1/0/0
 PTR productsearch.ubuntu.com. (82)
23:14:09.728442 IP packetflows.local.12125 > cdns01.comcast.net.domain: 56414+ PTR?
 15.2.0.10.in-addr.arpa. (40)
23:14:09.763628 IP cdns01.comcast.net.domain > packetflows.local.12125: 56414 NXDom
ain 0/0/0 (40)
23:14:09.863208 IP productsearch.ubuntu.com.https > packetflows.local.47860: Flags
[S.], seq 1389760001, ack 2046377879, win 65535, options [mss 1460], length 0
23:14:09.863298 IP packetflows.local.47860 > productsearch.ubuntu.com.https: Flags
[.], ack 1, win 29200, length 0
23:14:09.863432 IP cdns02.comcast.net.domain > packetflows.local.10745: 47145 1/0/0
 PTR productsearch.ubuntu.com. (82)
23:14:09.863465 IP packetflows.local > cdns02.comcast.net: ICMP packetflows.local u
dp port 10745 unreachable, length 118
23:14:09.864415 IP packetflows.local.47860 > productsearch.ubuntu.com.https: Flags
```


https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html

⚠ READ: Any knowledge and techniques presented here are for your learning purposes only. It is **ABSOLUTELY ILLEGAL** to apply the learned knowledge to others without proper consent/permission, and even then, you must check and comply with any regulatory restrictions and laws.

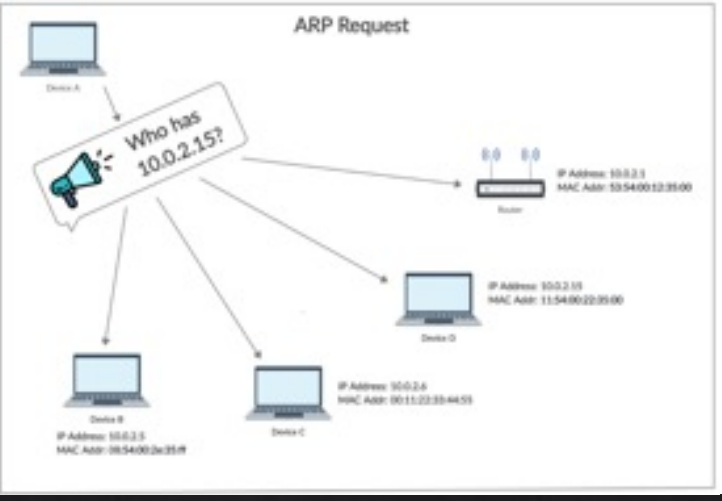
Sniffing - Wireshark

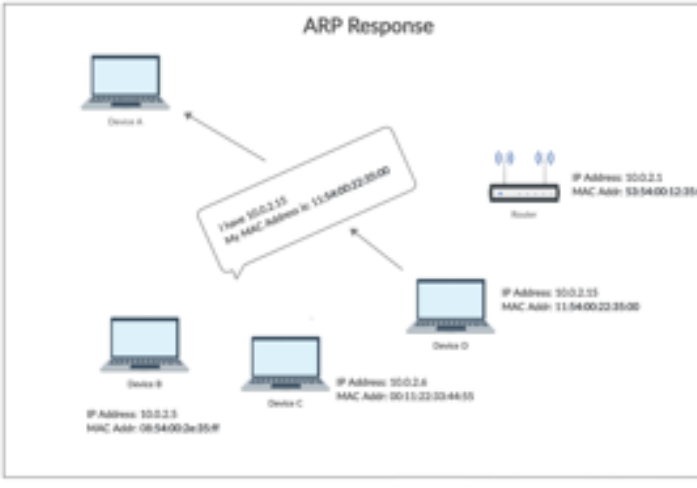
- Demo

https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/capture.pcap

Network Scanner

- We can use ARP (Address Resolution Protocol) to discover hosts in the network, which translates IP to MAC addresses.
- We are accustomed to use IP addresses to communicate, but computers use MAC address to communicate – we can use this fact to discover live IP addresses using ARP.
- By checking IP via ARP, we know the host at the IP address exists if we get a reply.





Network Scanner

- Demo

wget https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/network_scanner.py

Network Scanner

- It really is this simple!

```python
import scapy.all as scapy
import argparse

def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('-t', '--target', dest='target', help='Target IP Address/Adresses'
    options = parser.parse_args()

    #Check for errors i.e if the user does not specify the target IP Address
    #Quit the program if the argument is missing
    #While quitting also display an error message
    if not options.target:
        #Code to handle if interface is not specified
        parser.error("[-] Please specify an IP Address or Addresses, use --help for more i
    return options

def scan(ip):
    arp_req_frame = scapy.ARP(pdst = ip)
    broadcast_ether_frame = scapy.Ether(dst = "ff:ff:ff:ff:ff:ff")
    broadcast_ether_arp_req_frame = broadcast_ether_frame / arp_req_frame

    answered_list = scapy.srp(broadcast_ether_arp_req_frame, timeout = 1, verbose = False)
    result = []
    for i in range(0,len(answered_list)):
        client_dict = {"ip" : answered_list[i][1].psrc, "mac" : answered_list[i][1].hwsrc}
        result.append(client_dict)
    return result

def display_result(result):
    print("-----------------------------------------\nIP Address\tMAC Address\n-----------------
    for i in result:
        print("{}\t{}".format(i["ip"], i["mac"]))


options = get_args()
scanned_output = scan(options.target)
display_result(scanned_output)
```

Network Scanner

- We can scan and view other hosts in the network!

```
┌──(jin㉿kali)-[~/cits3006/lecture1]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:73:0d:97:14  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.67.3  netmask 255.255.255.0  broadcast 192.168.67.255
        ether 2e:06:ec:ba:8f:92  txqueuelen 1000  (Ethernet)
        RX packets 7  bytes 1326 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22  bytes 2551 (2.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(jin㉿kali)-[~/cits3006/lecture1]
└─$ sudo python3 network_scanner.py -t 192.168.67.0/24
[sudo] password for jin:
IP Address      MAC Address
192.168.67.1    be:d0:74:b2:00:64
192.168.67.6    e6:04:a6:2f:3c:54
```

Port Scanner

- Once we figure out the host we would like to exploit, the next step is to discover what services it has running
- Services expose selected port(s) to provide connections to others
- We can check whether the port is open or not using socket programming

Port Scanner

- Demo

wget https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/port_scanner.py

Port Scanner

- The socket function call takes the IP address to initialize the socket connection
- Once the port is specified, the connection attempt is made
- A response 0 indicates the port is open (i.e., can connect)

```
1 from socket import *
2 import argparse
3
4 def get_args():
5     parser = argparse.ArgumentParser()
6     parser.add_argument('-t', '--target', dest='target', help='Target IP Addres
7     options = parser.parse_args()
8
9     #Check for errors i.e if the user does not specify the target IP Address
10    #Quit the program if the argument is missing
11    #While quitting also display an error message
12    if not options.target:
13        #Code to handle if interface is not specified
14        parser.error("[-] Please specify an IP Address or Addresses, use --help
15    return options
16
17 def main():
18   if __name__ == '__main__':
19      target = get_args().target
20      t_IP = gethostbyname(target)
21      print ('Starting scan on host: ', t_IP)
22
23      for i in range(1, 1024):
24         s = socket(AF_INET, SOCK_STREAM)
25
26         conn = s.connect_ex((t_IP, i))
27         if(conn == 0) :
28            print ('Port %d: OPEN' % (i,))
29         s.close()
30
31 main()
```

Port Scanner

- We see that bunch of ports are open!
- We see familiar ports open:
  - FTP (21)
  - SSH (22)
  - Telnet (23)
  - SMTP (25)
  - DNS (53)
  - HTTP (80)

```
┌──(jin㉿kali)-[~/cits3006/lect1]
└─$ ls
network_scanner.py  port_scanner.py

┌──(jin㉿kali)-[~/cits3006/lect1]
└─$ sudo python3 port_scanner.py -t 192.168.67.6
Starting scan on host:  192.168.67.6
Port 21: OPEN
Port 22: OPEN
Port 23: OPEN
Port 25: OPEN
Port 53: OPEN
Port 80: OPEN
Port 111: OPEN
Port 139: OPEN
Port 445: OPEN
Port 512: OPEN
Port 513: OPEN
Port 514: OPEN

┌──(jin㉿kali)-[~/cits3006/lect1]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:88:15:98:61  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.67.7  netmask 255.255.255.0  broadcast 192.168.67.2
        inet6 fe80::b0d2:91ff:fe43:86e5  prefixlen 64  scopeid 0x20<link
```

More will be explored in Lab 1!