Share

Section 1: MITRE PRE-ATT&CK

INFOSEC Skills

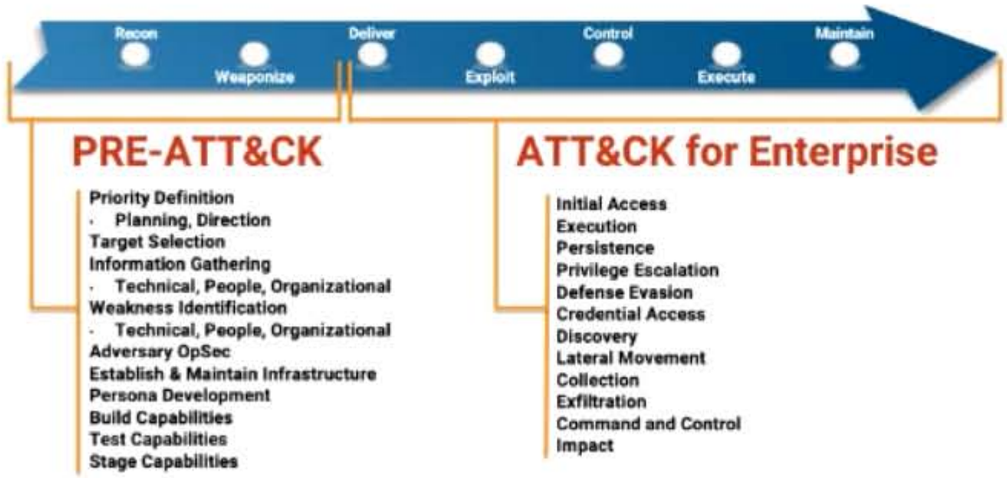
Hello and welcome to this course on MITRE PRE- ATTACK.

0:00 / 5:25

Introduction to PRE-ATT&CK

- MITRE PRE-ATT&CK matrix used to be its own standalone matrix
  - Contained a collection of Tactics and Techniques
  - Mapped to the Recon and Weaponize stages of the cyber kill chain

- Now, PRE-ATT&CK is the first two stages of the MITRE ATT&CK for the Enterprise framework
  - Reconnaissance
  - Resource Development



that covers many of the same stages of an attack,

INFOSEC Skills

PRE-ATT&CK: Reconnaissance

- The first stage of PRE-ATT&CK focuses on gathering target information from a variety of different sources:
  - Active Scanning
  - Gather Victim Host Information
  - Gather Victim Identity Information
  - Gather Victim Network Information
  - Gather Victim Org Information
  - Phishing for Information
  - Search Closed Sources
  - Search Open Technical Databases
  - Search Open Websites/Domains
  - Search Victim-Owned Websites

users that are associated
with this network.

PRE-ATT&CK: Resource Development

- The second stage of PRE-ATT&CK involves the attacker developing or acquiring the tools needed to perform their attack:
  - Acquire Infrastructure
  - Compromise Accounts
  - Compromise Infrastructure
  - Develop Capabilities
  - Establish Accounts
  - Obtain Capabilities

identified during the
reconnaissance stage.

INFOSEC Skills

Python for PRE-ATT&CK

- The Resource Development Tactic of PRE-ATT&CK largely occurs on the attacker's infrastructure
  - No interaction with target systems for defenders to detect
  - Depends heavily on the attacker's goals and resources

- This course explores two Techniques from the Reconnaissance Tactic of PRE-ATT&CK:
  - Active Scanning: Network Scanning
  - Search Open Technical Databases: DNS Exploration

INFOSEC Skills

an introduction to network scanning. Thank you.

Share

Section 2: Introduction to network scanning

INFOSEC Skills

Hello and welcome to this course in which we're

0:01 / 5:01

Network scanning

- Knowledge of a target network is vital for an attacker
  - Identification of potential target systems
  - Discovery of vulnerable applications

- Network scanning is one method of learning a target network architecture
  - Port scanning
  - Banner collection
  - Vulnerability scanning

is the banner collection
or banner grabbing.

```
ubuntu:~/Python_for_Cybersecurity/Part_2/2.1_Network_Scanning$ python
Python 3.9.1 (default, Dec 16 2020, 19:10:14)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> packet = rdpcap('http.cap')
>>> packet
<http.cap: TCP:41 UDP:2 ICMP:0 Other:0>
>>> p = packet[0]
>>> p.show()
```


its contents using the show
function built into scapy.

```
>>> packet
<http.cap: TCP:41 UDP:2 ICMP:0 Other:0>
>>> p = packet[0]
>>> p.show()
###[ Ethernet ]###
  dst       = fe:ff:20:00:01:00
  src       = 00:00:01:00:00:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 48
     id        = 3905
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x91eb
     src       = 145.254.160.237
     dst       = 65.208.228.223
     \options   \
###[ TCP ]###
        sport     = 3372
        dport     = http
        seq       = 951057939
        ack       = 0
        dataofs   = 7
        reserved  = 0
        flags     = S
        window    = 8760
        chksum    = 0xc30c
        urgptr    = 0
        options   = [('MSS', 1460), ('NOP', None), ('NOP', None), ('SACKOK', b'')]

>>> p = packet[0]
```

we need to look at the third
or the fourth packet in

```
  src       = 00:00:01:00:00:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 519
     id        = 3909
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x9010
     src       = 145.254.160.237
     dst       = 65.208.228.223
     \options   \
###[ TCP ]###
        sport     = 3372
        dport     = http
        seq       = 951057940
        ack       = 290218380
        dataofs   = 5
        reserved  = 0
        flags     = PA
        window    = 9660
        chksum    = 0xa958
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'GET /download.html HTTP/1.1\r\nHost: www.ethereal.com\r\nUser-Agent: Mozilla/5.0 (Windows;
 U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\nAccept: text/xml,application/xml,applicat
tml;q=0.9,text/plain;q=0.8,image/p                                                    cept-Language:
cept-Encoding: gzip,deflate\r\nAcc                                                    nKeep-Alive: 3
ep-alive\r\nReferer: http://www.ethe                                                  ereal.com/development.html\r\n\r\n'
>>>
```

To demonstrate what we can do with scapy, for example,

```
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 519
     id        = 3909
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x9010
     src       = 145.254.160.237
     dst       = 65.208.228.223
     \options   \
###[ TCP ]###
        sport     = 3372
        dport     = http
        seq       = 951057940
        ack       = 290218380
        dataofs   = 5
        reserved  = 0
        flags     = PA
        window    = 9660
        chksum    = 0xa958
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'GET /download.html HTTP/1.1\r\nHost: www.ethereal.com\r\nUser-Agent: Mozilla/5.0 (Windows;
 U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/h
tml;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1\r\nAccept-Language:
cept-Encoding: gzip,deflate\r\nAccep                                    \nKeep-Alive: 3
ep-alive\r\nReferer: http://www.ethe

>>> p[TCP].dport = 8080
>>> p.show()
```

hit enter and then show the contents of the packet again.

```
  src       = 00:00:01:00:00:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 519
     id        = 3909
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x9010
     src       = 145.254.160.237
     dst       = 65.208.228.223
     \options   \
###[ TCP ]###
        sport     = 3372
        dport     = 8045
        seq       = 951057940
        ack       = 290218380
        dataofs   = 5
        reserved  = 0
        flags     = PA
        window    = 9660
        chksum    = 0xa958
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'GET /download.html HTTP/1.1\r\nHost: www.ethereal.com\r\nUser-Agent: Mozilla/5.0 (Windows;
 U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\nAccept: text/xml,application/xml,applicat
tml;q=0.9,text/plain;q=0.8,image/png,imag                                    \r\nAccept-Language:
cept-Encoding: gzip,deflate\r\nAccept-Cha                                    =0.7\r\nKeep-Alive: 3
ep-alive\r\nReferer: http://www.etherea                                      .com/development.html\r\n\r\n'
>>>
```

that scapy will show the actual port number.

```
          load      = 'GET /download.html HTTP/1.1\r\nHost: www.ethereal.com\r\nUser-Agent:
 U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\nAccept: text/xml,application/xml,applic
tml;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1\r\nAccept-Language: en-us,en;q=0.5\r\nAc
cept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\nKeep-Alive: 300\r\nConnection: ke
ep-alive\r\nReferer: http://www.ethereal.com/development.html\r\n\r\n'

>>> p = IP()/TCP()
>>> p.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = http
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

>>>
```


```
    flags     = S
    window    = 8192
    chksum    = None
    urgptr    = 0
    options   = []

>>> p[TCP].dport = 35
>>> p.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = 35
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

>>>
```

take a look we see that
updates, et cetera,

```
>>> p = IP(dst=8.8.8.8)/TCP(dport=53)
  File "<stdin>", line 1
    p = IP(dst=8.8.8.8)/TCP(dport=53)
                  ^
SyntaxError: invalid syntax
>>> p = IP(dst="8.8.8.8")/TCP(dport=53)
>>> p.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 172.31.18.163
  dst       = 8.8.8.8
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = domain
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

>>>
```

and p.show, we now see that it's going to domain.

```
>>> p = IP(dst=8.8.8.8)/TCP(dport=53)
  File "<stdin>", line 1
    p = IP(dst=8.8.8.8)/TCP(dport=53)
                   ^
SyntaxError: invalid syntax
>>> p = IP(dst="8.8.8.8")/TCP(dport=53)
>>> p.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 172.31.18.163
  dst       = 8.8.8.8
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = domain
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

>>> p = IP(dst="8.8.8.8")/UDP(dport=53)/DNS()
>>> p.show()
```

```
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 172.31.18.163
  dst       = 8.8.8.8
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = domain
     len       = None
     chksum    = None
###[ DNS ]###
        id        = 0
        qr        = 0
        opcode    = QUERY
        aa        = 0
        tc        = 0
        rd        = 1
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 0
        ancount   = 0
        nscount   = 0
        arcount   = 0
        qd        = None
        an        = None
        ns        = None
        ar        = None
>>>
```

and it's created a default
DNS packet for us.

```
ubuntu:~/Python_for_Cybersecurity/Part_2/2.1_Network_Scanning$ nano PortScan.py
ubuntu:~/Python_for_Cybersecurity/Part_2/2.1_Network_Scanning$ python PortScan.py
Traceback (most recent call last):
  File "PortScan.py", line 19, in <module>
    SynScan(host)
  File "PortScan.py", line 6, in SynScan
    ans,unans = sr(IP(dst=host)/TCP(sport=5555,dport=ports,flags="S"),timeout=2,verbose=0)
  File "/home/ubuntu/.local/lib/python2.7/site-packages/scapy/sendrecv.py", line 509, in sr
    iface=iface, nofilter=nofilter)
  File "/home/ubuntu/.local/lib/python2.7/site-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))  # noqa: E501
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
ubuntu:~/Python_for_Cybersecurity/Part_2/2.1_Network_Scanning$ sudo python PortScan.py
Open ports at 8.8.8.8:
53
443
DNS Server at 8.8.8.8
ubuntu:~/Python_for_Cybersecurity/Part_2/2.1_Network_Scanning$
```

just some simple
scanners that we're

Share

Section 3: Introduction to open technical databases

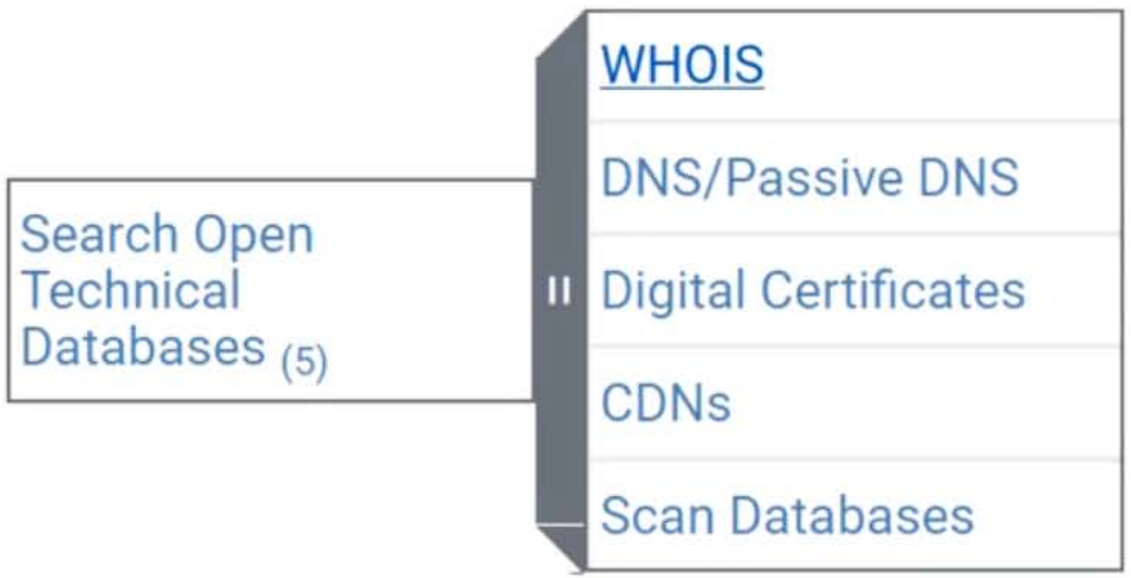
INFOSEC Skills

Hello, and welcome to this course

0:01 / 4:37

Open technical databases

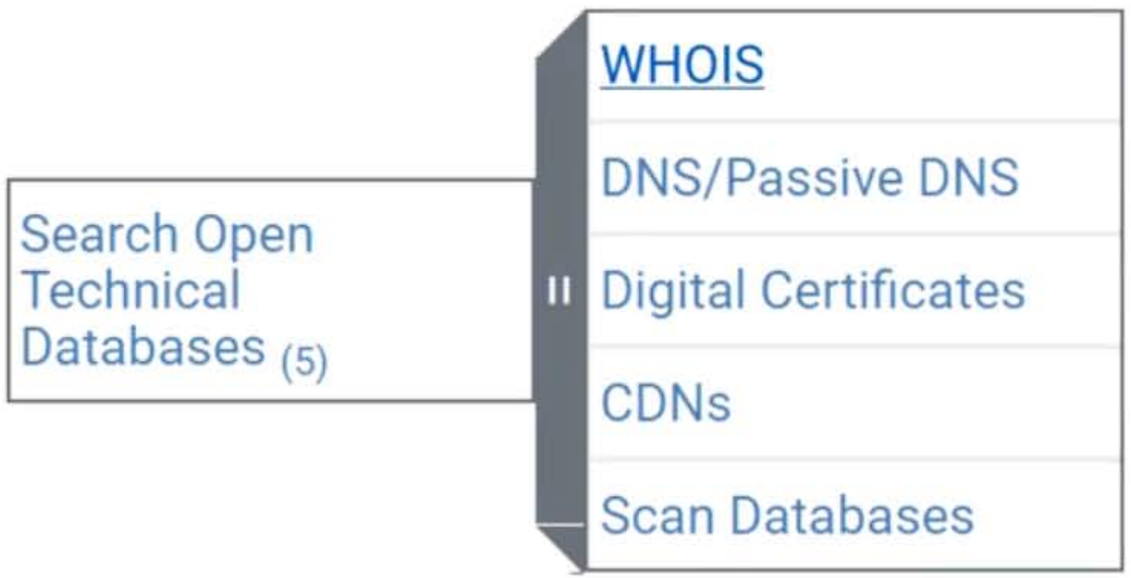
- Open-source intelligence (OSINT) is a trove of useful data regarding an organization and its systems. Examples of these datasets include:
  - WHOIS: WHOIS records may include data about owners of websites, system administrators, etc.
  - DNS: DNS maps domain names to IP addresses
  - CDNs: CDNs store cached content for an organization's websites



A CDN is designed to

Open technical databases

- Open-source intelligence (OSINT) is a trove of useful data regarding an organization and its systems. Examples of these datasets include:
  - WHOIS: WHOIS records may include data about owners of websites, system administrators, etc.
  - DNS: DNS maps domain names to IP addresses
  - CDNs: CDNs store cached content for an organization's websites



deliver content from a particular organization's website.

```
ubuntu:~/Python_for_Cybersecurity/Part_2/2.2_DNS_Exploration$ python DNSExploration.py
www.google.com
172.217.4.36
Domain Names: ['lga15s46-in-f36.1e100.net']
www4.google.com
172.217.8.174
Domain Names: ['ord37s08-in-f14.1e100.net']
www5.google.com
172.217.8.164
Domain Names: ['ord37s08-in-f4.1e100.net']
www6.google.com
172.217.8.196
Domain Names: ['ord37s09-in-f4.1e100.net']
www9.google.com
172.217.4.46
Domain Names: ['ord38s18-in-f14.1e100.net']
mail.google.com
172.217.9.37
Domain Names: ['ord38s08-in-f5.1e100.net']
blog.google.com
172.217.4.73
Domain Names: ['lga15s47-in-f73.1e100.net']
ns.google.com
216.239.32.10
Domain Names: ['ns1.google.com']
ns1.google.com
216.239.32.10
Domain Names: ['ns1.google.com']
ns2.google.com
216.239.34.10
Domain Names: ['ns2.google.com']
ns3.google.com
216.239.36.10
Domain Names: ['ns3.google.com']
ns4.google.com
216.239.38.10
```


And so in this particular case, it's returning an IP address of 172.217.4.36.