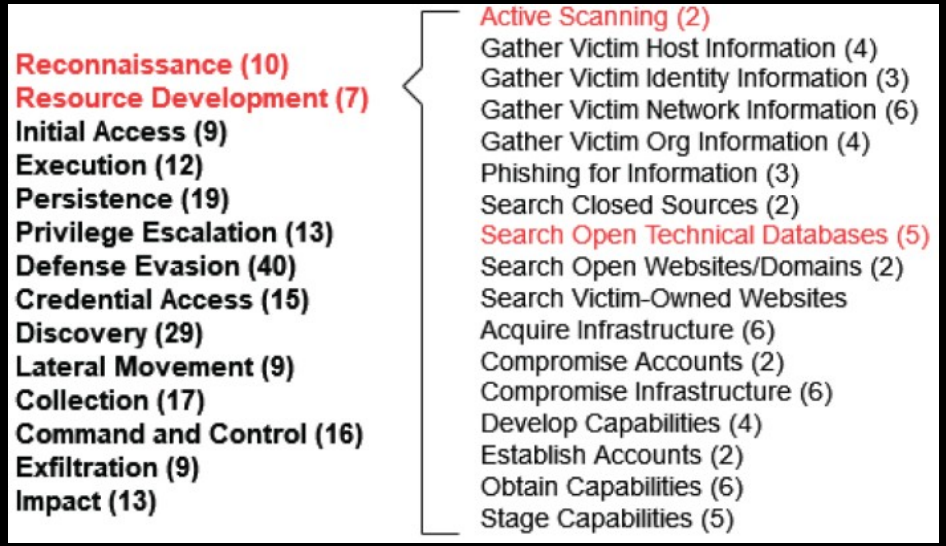
Chapter 1: Fulfilling Pre-ATT&CK Objectives

What is MITRE Pre-ATT&CK

- Originally a standalone matrix
- Now merged into Reconnaissance and Resource Development (2020)
- Focuses on attacker preparation before exploitation

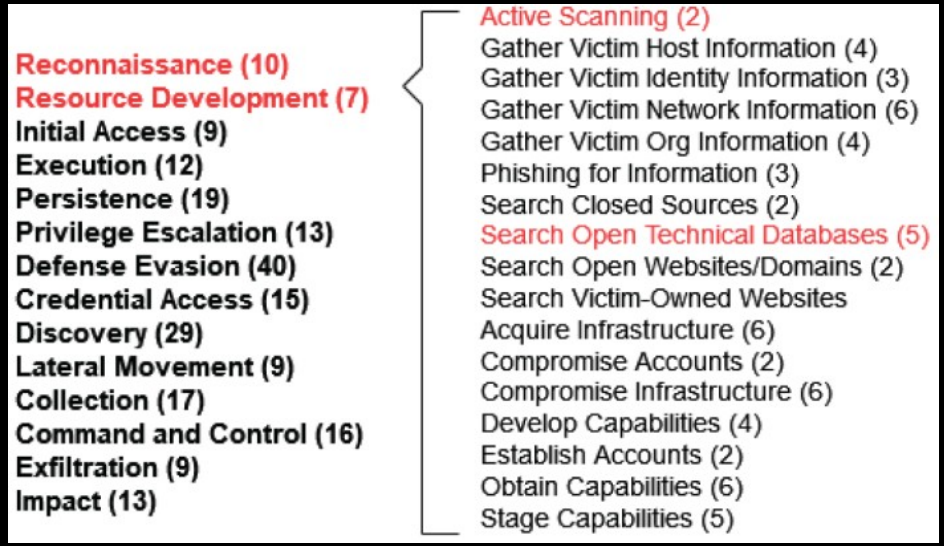
What is MITRE Pre-ATT&CK

- Let’s talk about this process
- Foundation of cyberattacks



Importance of Reconnaissance

- Attackers gather intelligence before exploitation
- Defenders must anticipate and detect reconnaissance



Why Python for Scripting?

- Popular and widely used
- Easy to learn (you should be familiar with Python) and implement
- Powerful libraries (scapy, dnspython, dnslib)

Active vs Passive Reconnaissance

- Active: Direct interaction with targets (scanning)
- Passive: Collecting publicly available information
- Active Recon is more detectable but reveals deeper insights

Active Scanning in MITRE ATT&CK

- Identifies active IPs, services, vulnerabilities
- Common techniques: Port and Vulnerability Scans
- MITRE categorizes it under Reconnaissance

SYN Scan Theory

- Uses TCP 3-way handshake
- SYN → SYN/ACK = Open Port
- SYN → RST/ACK or Timeout = Closed/Filtered

Python Example: PortScan.py

- Implements SYN and DNS scans
- Uses scapy for packet crafting
- Scans common ports (25, 80, 53, 443, 445, 8080, 8443)

DNS Scan with scapy

- Checks if DNS server is active on port 53
- Uses DNS queries (qd = DNSQR)
- Quick way to identify DNS servers in target network

Demo Output

- Example: 8.8.8.8 (Google DNS)
- Open ports: 53, 443
- DNS Server at 8.8.8.8

Introducing HoneyScan

- Defensive tool to mislead attackers
- Makes closed ports look open, open look closed
- Redirects traffic to honeypots

How HoneyScan Works

- Monitors traffic with sniff()
- Analyzes packets with analyzePackets()
- Crafts deceptive responses (RST/ACK, SYN/ACK)

Limitation of HoneyScan

- Race conditions (legit system vs HoneyScan)
- Best run inline on firewall/IPS
- Still useful to delay and trap attackers

Passive Reconnaissance

- No direct interaction with target
- Takes advantage of public data (DNS, WHOIS, etc.)
- Less detectable than active scanning

DNS as a Goldmine

- Maps domain names to IPs
- Subdomain enumeration reveals services
- Reverse lookups reveal hidden relationships

Python Example: DNSExploration.py

- Uses dnspython library
- Performs subdomain lookups
- Performs reverse DNS queries

Default Subdomains Table

- www, mail, vpn, ftp, admin, smtp
- Common naming conventions reveal purpose
- HostSearch brute-force with dictionary + numbers

Default Subdomains Table

- Some organization might use slight variations e.g.
- ns1.example.com
- ns2.example.com

www	secure	email
mail	vpn	cloud
remote	dns	owa
blog	ftp	admin
webmail	test	cdn
server	portal	api
ns	host	exchange
smtp	support	mysql
pop	dev	wiki
imap	web	cpanel
admin	mx	

Introducing HoneyResolver

- DNS server deception tool
- Returns real IPs for valid subdomains
- Redirects others to honeypot IP

How HoneyResolver Works

- Implements Resolver class in dnslib
- Valid subdomain = real IP
- Invalid subdomains = honeypot

Key Takeaways

- Reconnaissance = Foundation of Cyberattacks
- Python enables both offensive and defensive automation
- HoneyScan and HoneyResolver provide deception capabilities

Going Above and Beyound

- The SYNScan function in PortScan.py currently checks only if a port is open, based on if it returns a SYN/ACK packet. Modify the code to differentiate between closed ports (which return a RST) and ports filtered by a firewall (which return nothing).

Going Above and Beyound

- Currently, PortScan.py implements SYN and DNS scans. Modify it to include additional types of scans, such as ACK and XMAS scans.

Going Above and Beyound

- Revise DNSExploration.py to group results by IP address rather than domain name. This helps to identify systems with multiple functions within an organization.

Going Above and Beyound

- Currently HoneyResolver.py makes it easy to differentiate real and fake results because all fake results resolve to the same IP address. Modify the code to only resolve certain fake subdomains with unique IP addresses assigned to each.

Going Above and Beyound

- HoneyResolver.py only sends responses containing A records, which are inappropriate for some requests. Extend the code to send the correct type of record for each request.