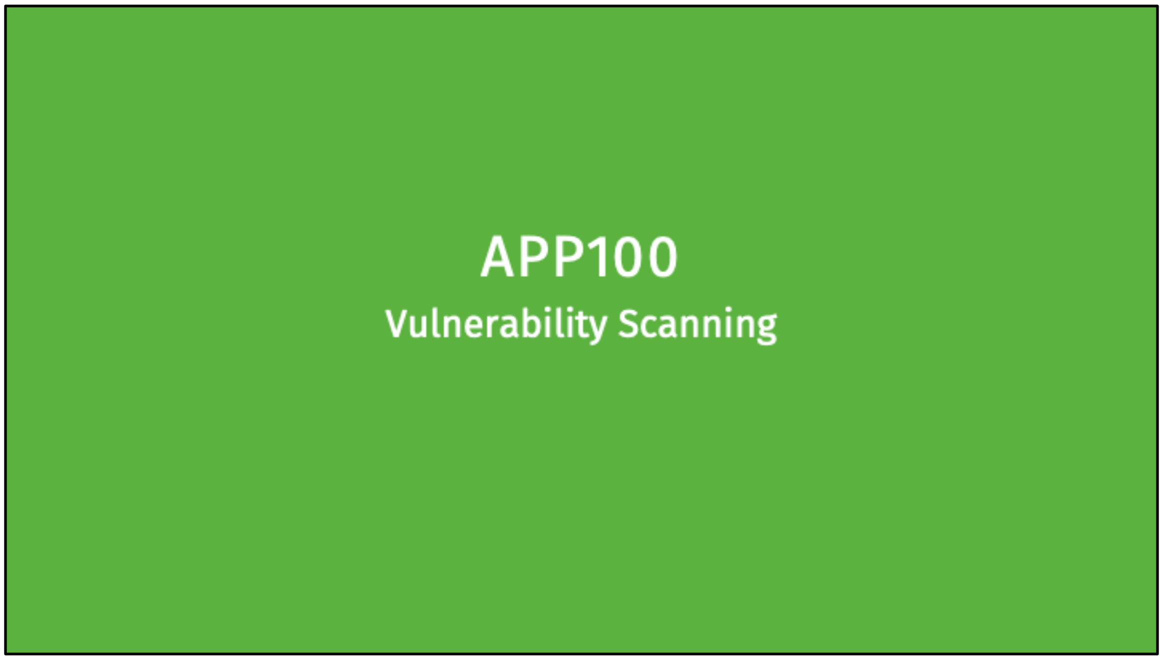
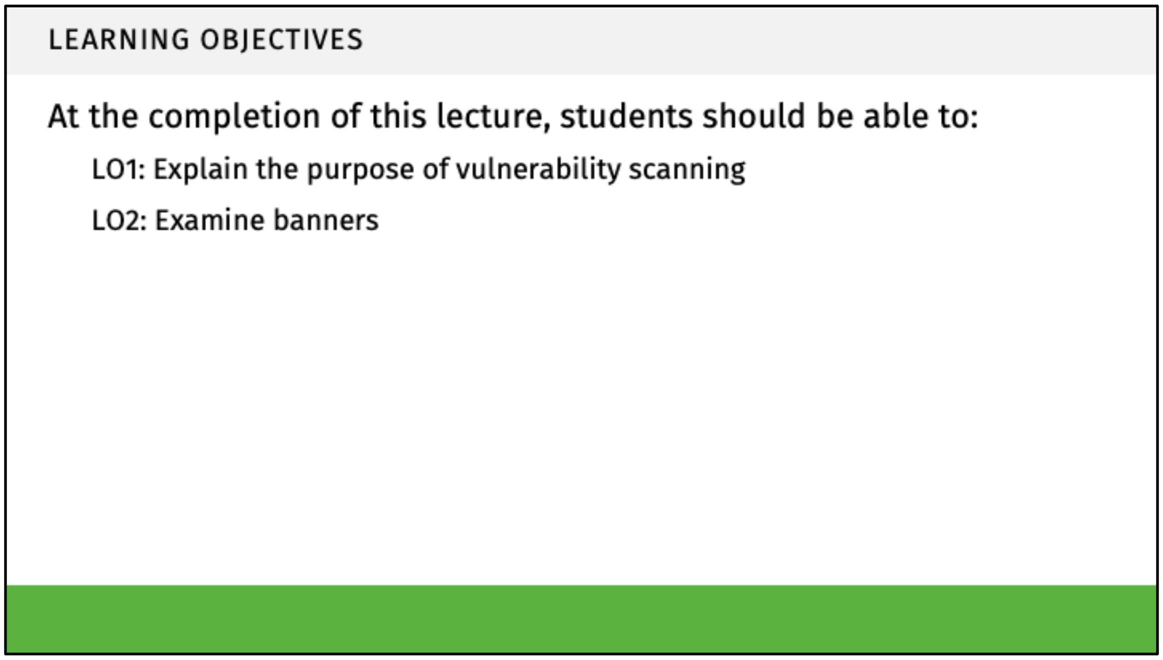


APP100

Vulnerability Scanning



LEARNING OBJECTIVES

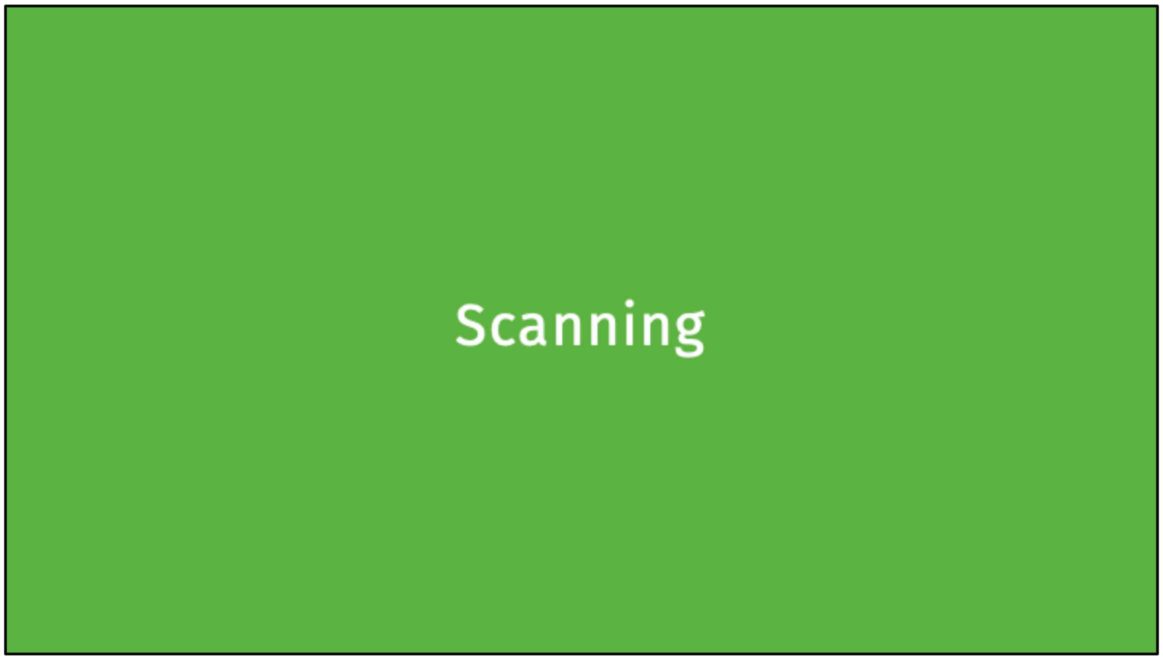
At the completion of this lecture, students should be able to:

- LO1: Explain the purpose of vulnerability scanning
- LO2: Examine banners

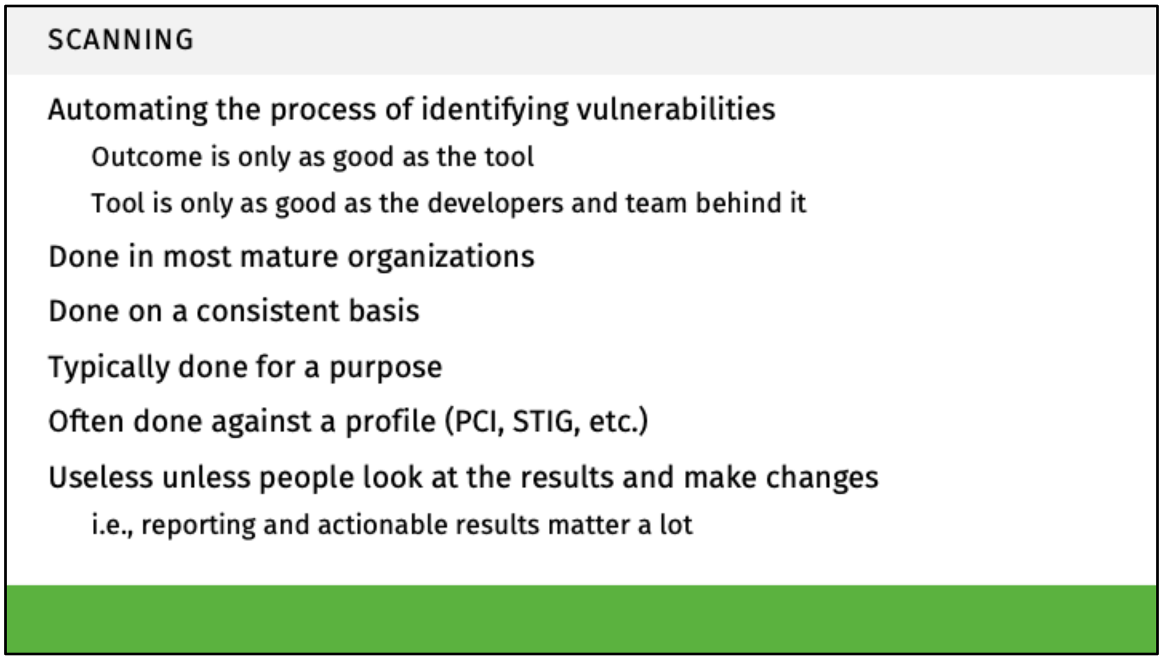
VULNERABILITIES

Weaknesses

- In code/software
- In processes
- In people
- In implementation

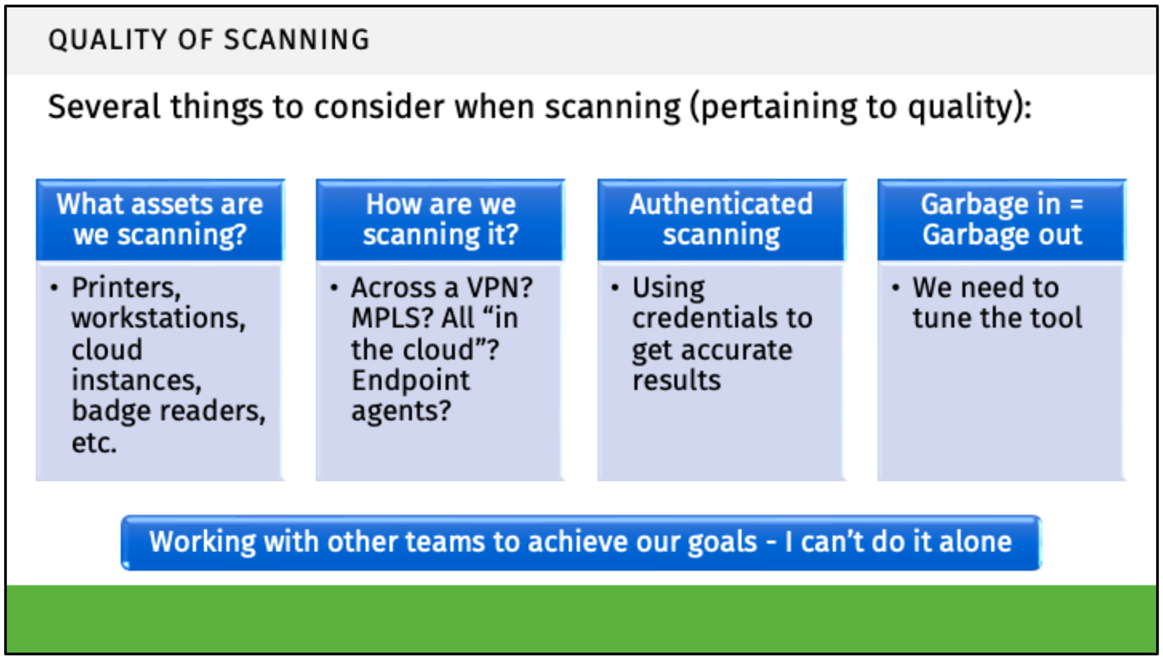


Scanning



SCANNING

- Automating the process of identifying vulnerabilities
  - Outcome is only as good as the tool
  - Tool is only as good as the developers and team behind it
- Done in most mature organizations
- Done on a consistent basis
- Typically done for a purpose
- Often done against a profile (PCI, STIG, etc.)
- Useless unless people look at the results and make changes
  - i.e., reporting and actionable results matter a lot



QUALITY OF SCANNING

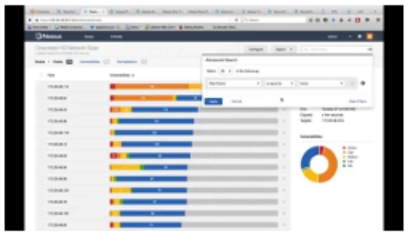
Several things to consider when scanning (pertaining to quality):

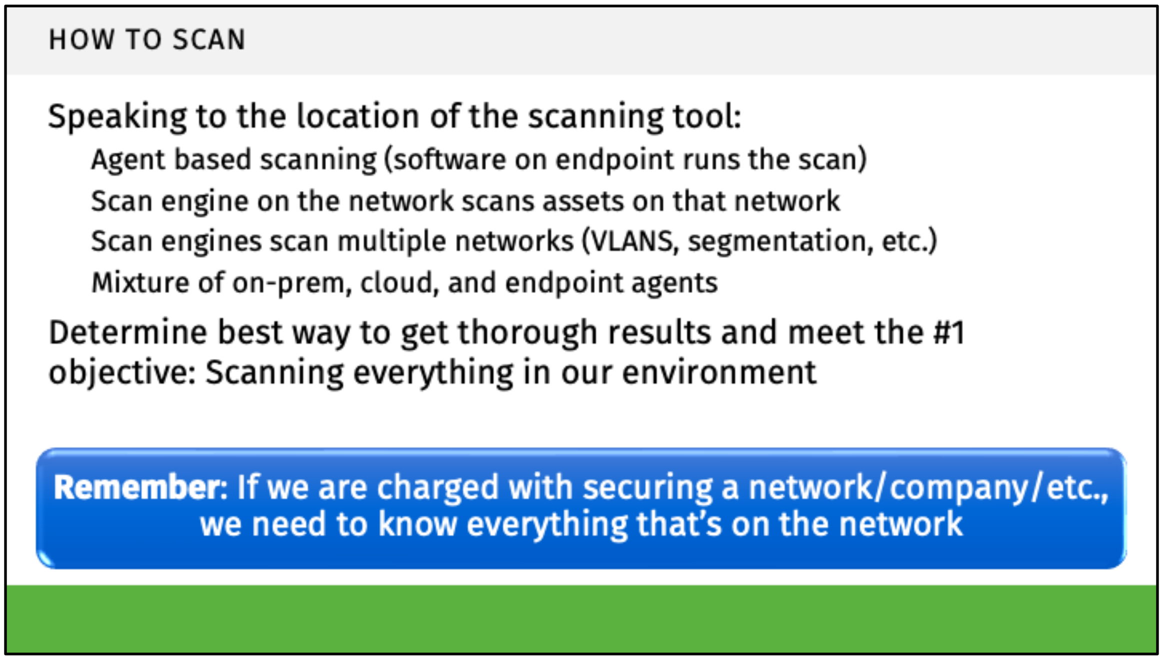
What assets are we scanning?	How are we scanning it?	Authenticated scanning	Garbage in = Garbage out
• Printers, workstations, cloud instances, badge readers, etc.	• Across a VPN? MPLS? All "in the cloud"? Endpoint agents?	• Using credentials to get accurate results	• We need to tune the tool

Working with other teams to achieve our goals - I can't do it alone

WHAT DO WE SCAN?

- Everything with an IP address
  - Every Server
  - Every workstation
  - Every endpoint
  - Every …
- If it's on our network, we need to scan it
  - Otherwise, we could have an unknown vulnerability/exposure
- How do we scan a remote user working from home?





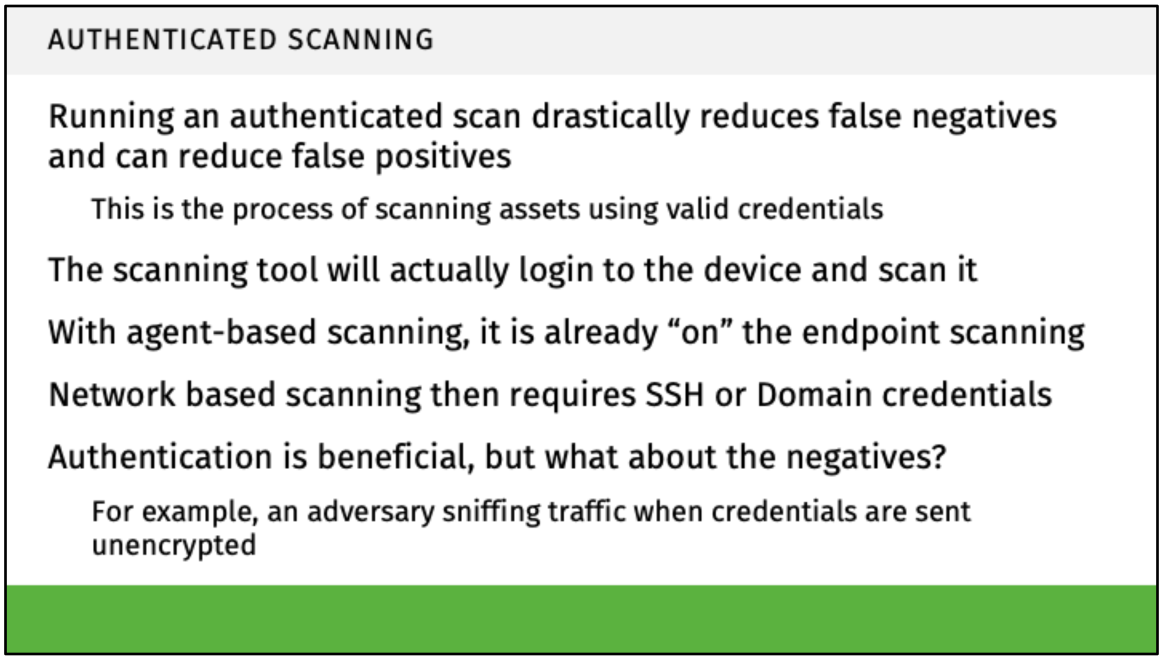
HOW TO SCAN

Speaking to the location of the scanning tool:

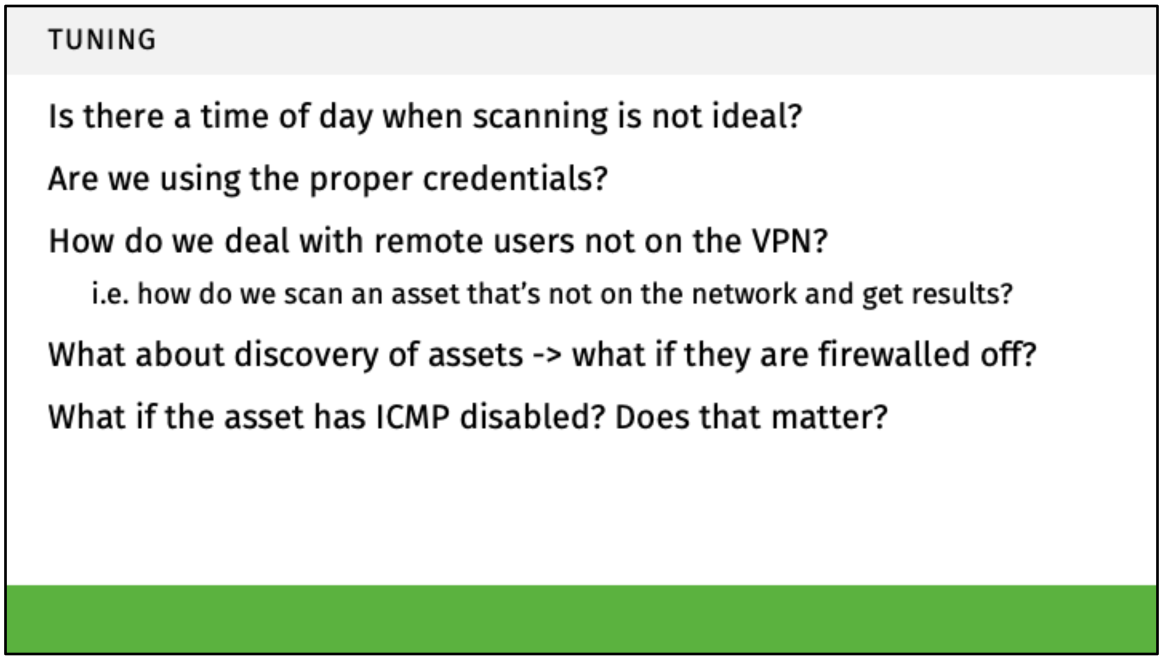
- Agent based scanning (software on endpoint runs the scan)
- Scan engine on the network scans assets on that network
- Scan engines scan multiple networks (VLANS, segmentation, etc.)
- Mixture of on-prem, cloud, and endpoint agents

Determine best way to get thorough results and meet the #1 objective: Scanning everything in our environment

Remember: If we are charged with securing a network/company/etc., we need to know everything that's on the network


AUTHENTICATED SCANNING

- Running an authenticated scan drastically reduces false negatives and can reduce false positives
  - This is the process of scanning assets using valid credentials
- The scanning tool will actually login to the device and scan it
- With agent-based scanning, it is already “on” the endpoint scanning
- Network based scanning then requires SSH or Domain credentials
- Authentication is beneficial, but what about the negatives?
  - For example, an adversary sniffing traffic when credentials are sent unencrypted



TUNING

Is there a time of day when scanning is not ideal?

Are we using the proper credentials?

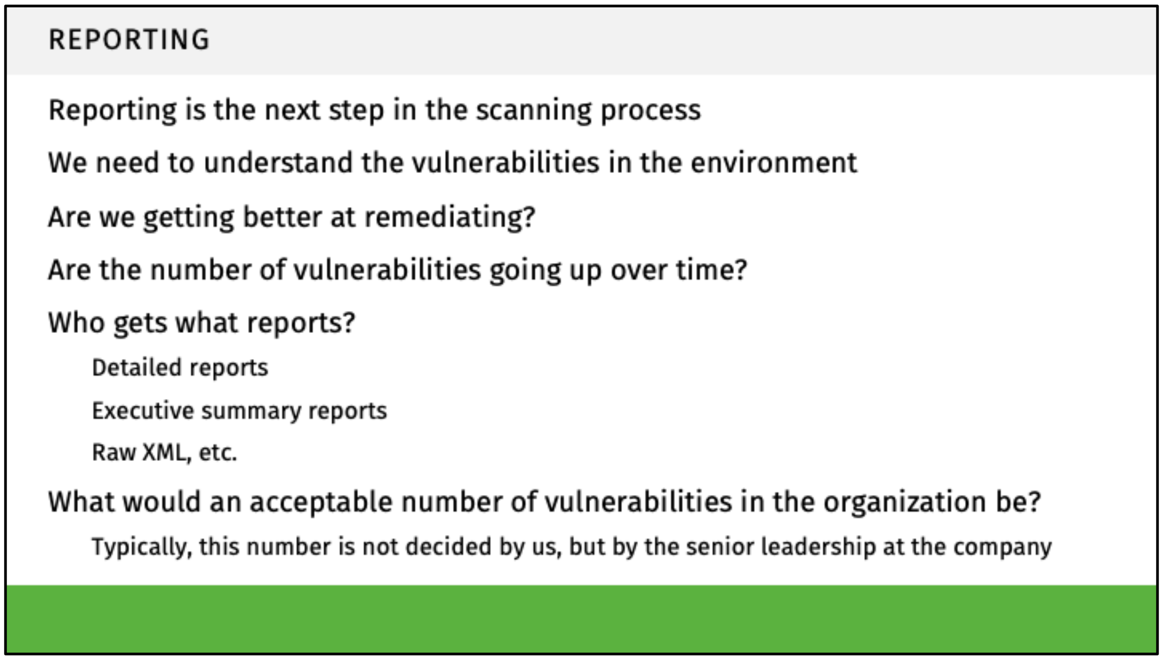
How do we deal with remote users not on the VPN?

- i.e. how do we scan an asset that’s not on the network and get results?

What about discovery of assets -> what if they are firewalled off?

What if the asset has ICMP disabled? Does that matter?

Reporting

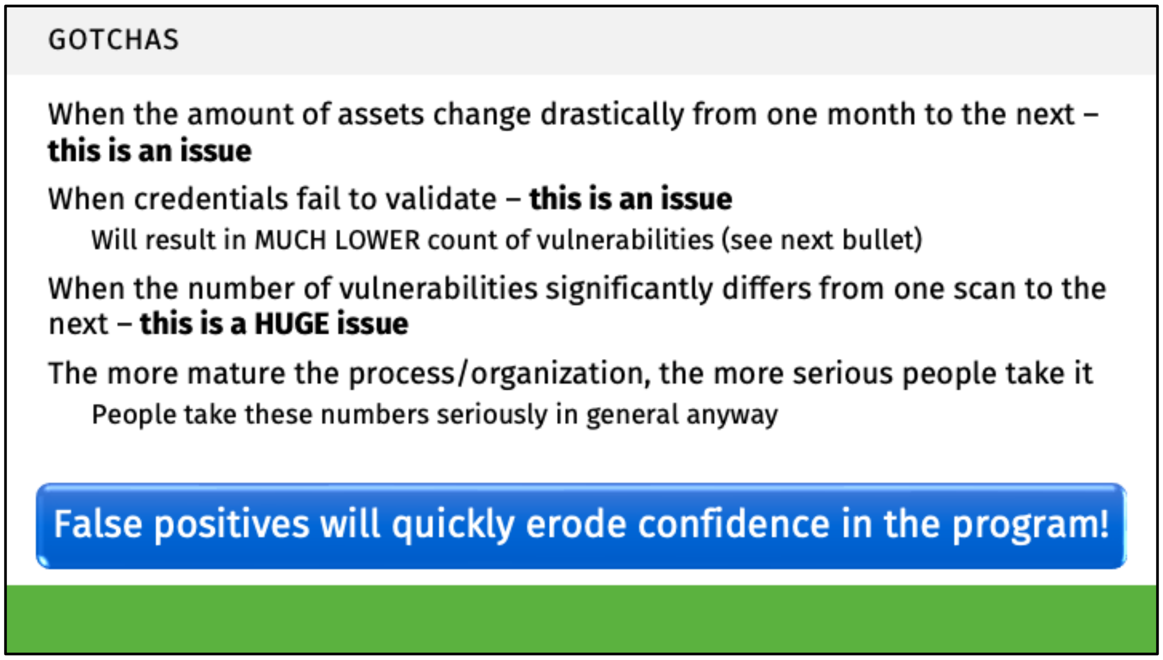


REPORTING

- Reporting is the next step in the scanning process
- We need to understand the vulnerabilities in the environment
- Are we getting better at remediating?
- Are the number of vulnerabilities going up over time?
- Who gets what reports?
  - Detailed reports
  - Executive summary reports
  - Raw XML, etc.
- What would an acceptable number of vulnerabilities in the organization be?
  - Typically, this number is not decided by us, but by the senior leadership at the company

METRICS

- Metrics are essentially results over time (but with graphs)
- Key areas of interest to track:
  - How many assets were scanned
  - How many vulnerabilities were discovered
  - Breaking down assets into groups of ownership
  - Reporting all of this on a regular basis
- Metrics are reported to management



GOTCHAS

When the amount of assets change drastically from one month to the next – **this is an issue**

When credentials fail to validate – **this is an issue**

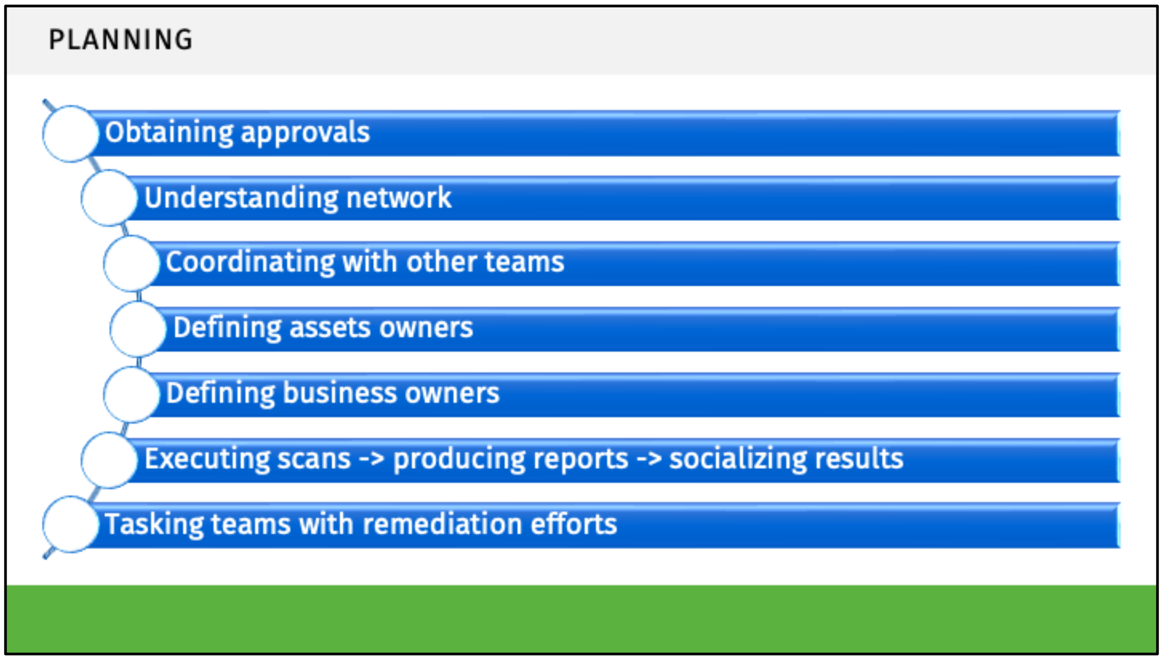
- Will result in MUCH LOWER count of vulnerabilities (see next bullet)

When the number of vulnerabilities significantly differs from one scan to the next – **this is a HUGE issue**

The more mature the process/organization, the more serious people take it

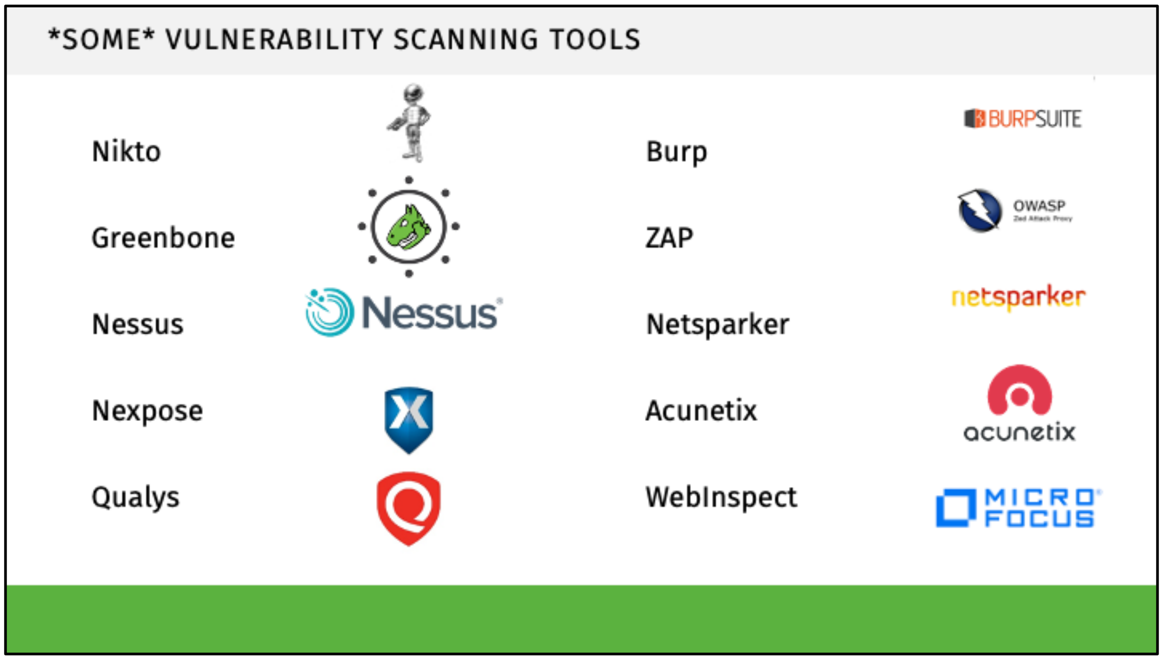
- People take these numbers seriously in general anyway

False positives will quickly erode confidence in the program!



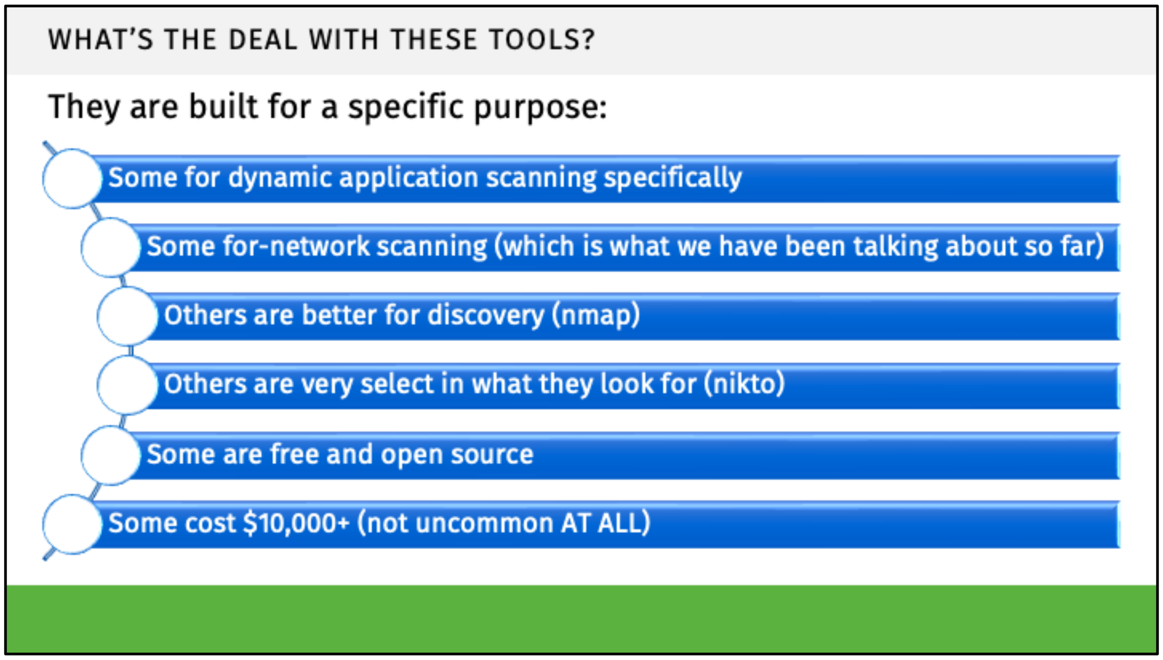
PLANNING

- Obtaining approvals
- Understanding network
- Coordinating with other teams
- Defining assets owners
- Defining business owners
- Executing scans -> producing reports -> socializing results
- Tasking teams with remediation efforts



SOME VULNERABILITY SCANNING TOOLS

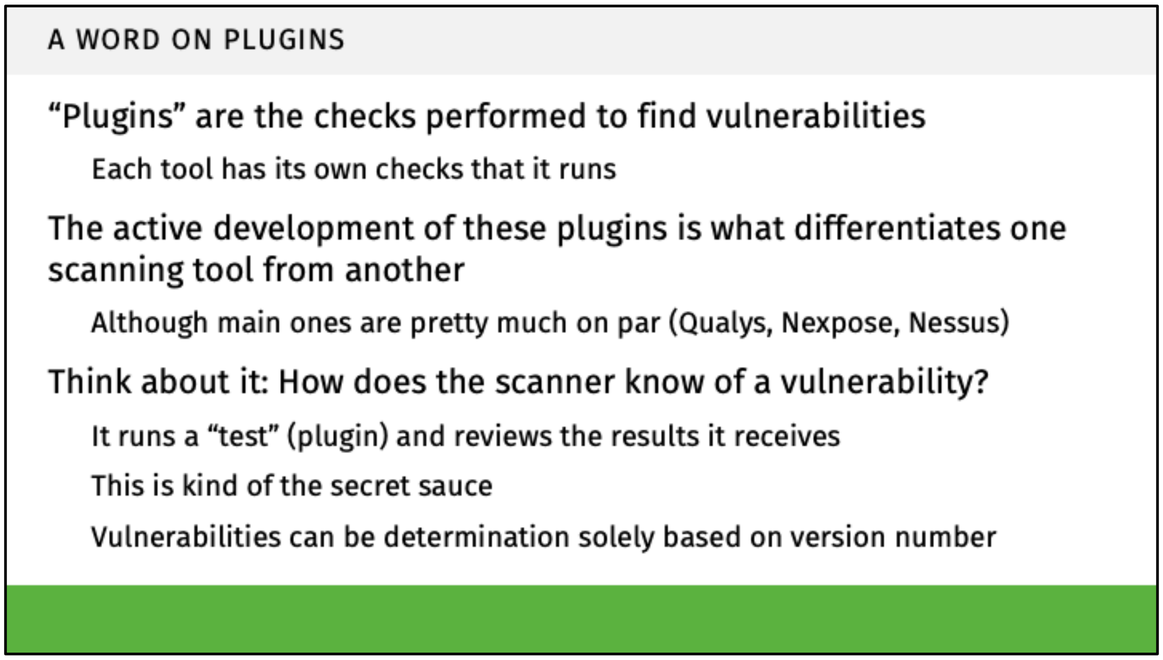
Nikto	Burp
Greenbone	ZAP
Nessus	Netsparker
Nexpose	Acunetix
Qualys	WebInspect



WHAT'S THE DEAL WITH THESE TOOLS?

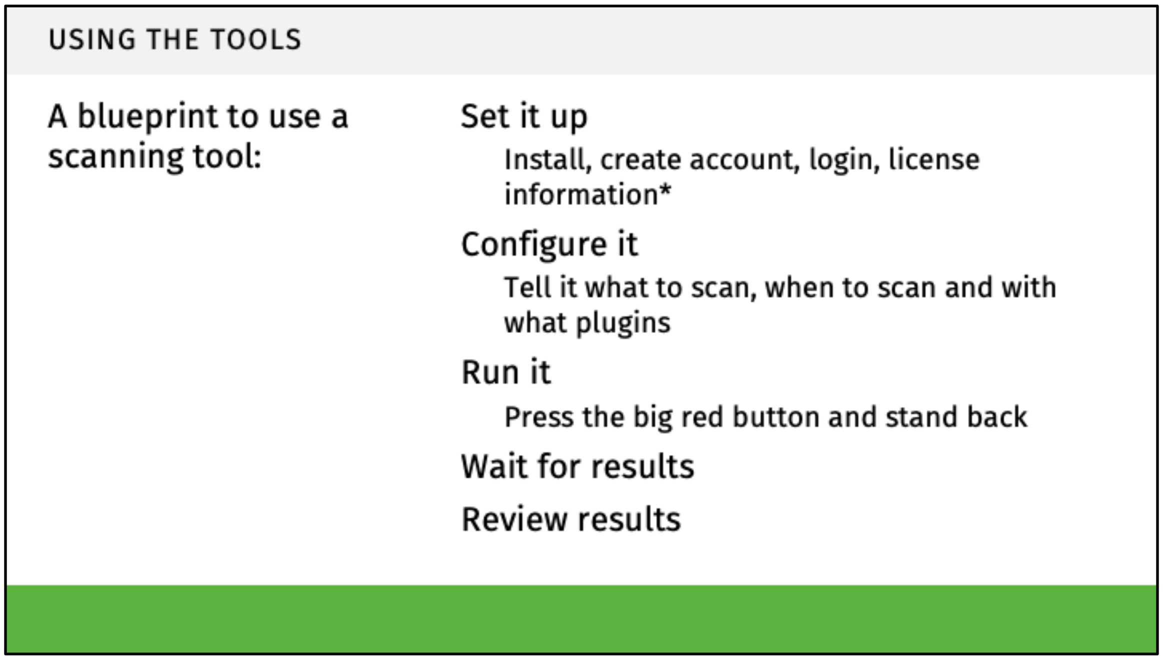
They are built for a specific purpose:

- Some for dynamic application scanning specifically
- Some for-network scanning (which is what we have been talking about so far)
- Others are better for discovery (nmap)
- Others are very select in what they look for (nikto)
- Some are free and open source
- Some cost $10,000+ (not uncommon AT ALL)



A WORD ON PLUGINS

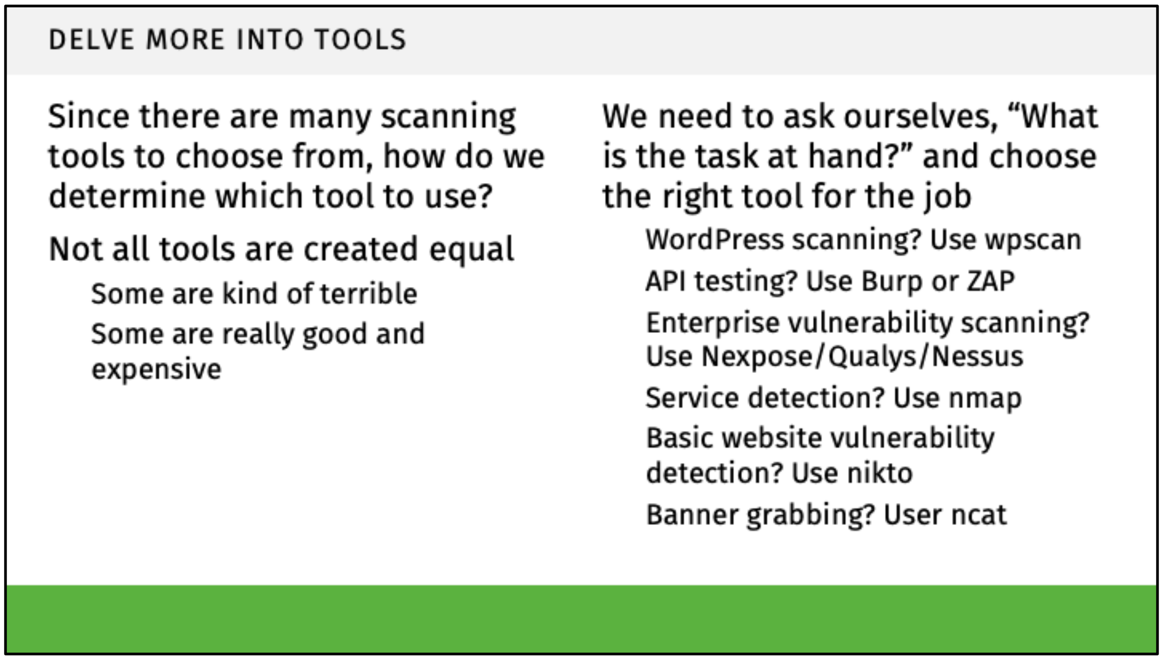
- "Plugins" are the checks performed to find vulnerabilities
  - Each tool has its own checks that it runs
- The active development of these plugins is what differentiates one scanning tool from another
  - Although main ones are pretty much on par (Qualys, Nexpose, Nessus)
- Think about it: How does the scanner know of a vulnerability?
  - It runs a "test" (plugin) and reviews the results it receives
  - This is kind of the secret sauce
  - Vulnerabilities can be determination solely based on version number



USING THE TOOLS

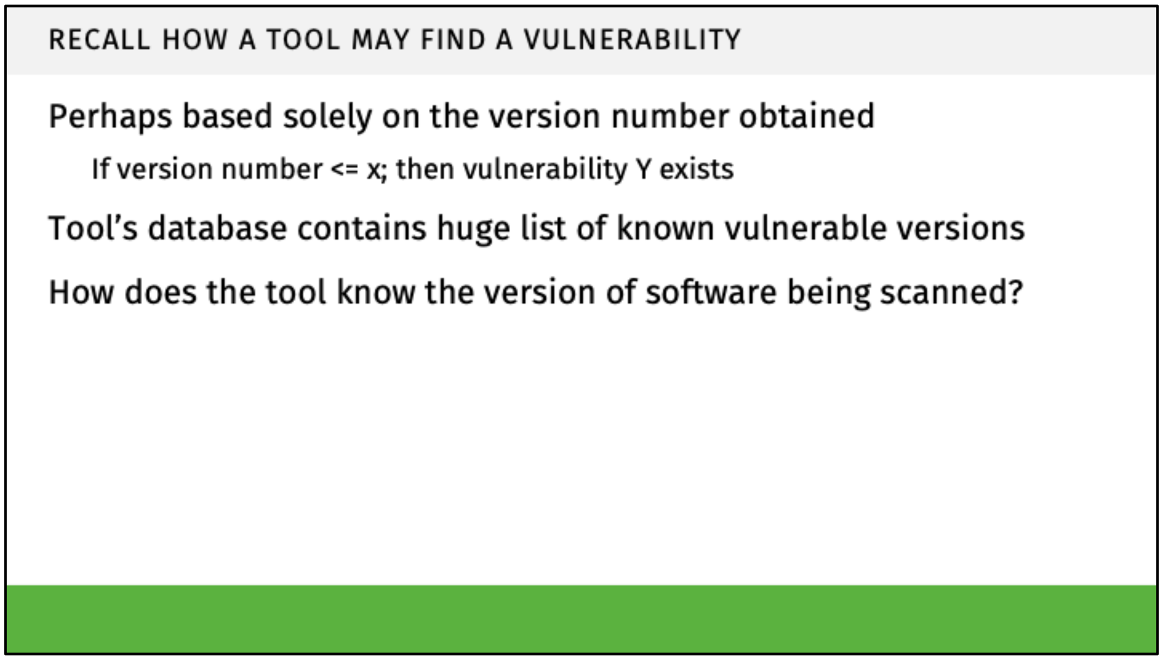
A blueprint to use a scanning tool:

- Set it up
  - Install, create account, login, license information*
- Configure it
  - Tell it what to scan, when to scan and with what plugins
- Run it
  - Press the big red button and stand back
- Wait for results
- Review results



DELVE MORE INTO TOOLS

- Since there are many scanning tools to choose from, how do we determine which tool to use?
- Not all tools are created equal
  - Some are kind of terrible
  - Some are really good and expensive
- We need to ask ourselves, “What is the task at hand?” and choose the right tool for the job
  - WordPress scanning? Use wpscan
  - API testing? Use Burp or ZAP
  - Enterprise vulnerability scanning? Use Nexpose/Qualys/Nessus
  - Service detection? Use nmap
  - Basic website vulnerability detection? Use nikto
  - Banner grabbing? User ncat



RECALL HOW A TOOL MAY FIND A VULNERABILITY

Perhaps based solely on the version number obtained

- If version number <= x; then vulnerability Y exists

Tool’s database contains huge list of known vulnerable versions

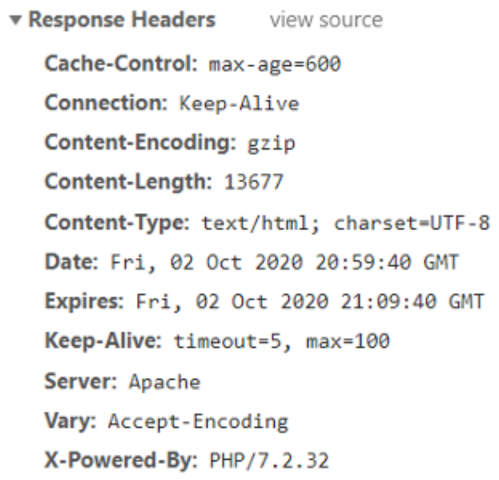
How does the tool know the version of software being scanned?

Banners

BANNERS

A banner in this context is essentially the response from a given service that conveys its version/information

We can see this on a typical website via HTTP response headers:



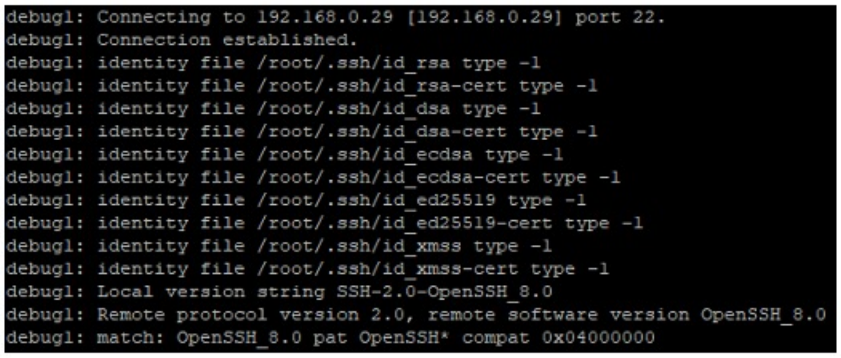
▾ Response Headers view source

```
Cache-Control: max-age=600
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 13677
Content-Type: text/html; charset=UTF-8
Date: Fri, 02 Oct 2020 20:59:40 GMT
Expires: Fri, 02 Oct 2020 21:09:40 GMT
Keep-Alive: timeout=5, max=100
Server: Apache
Vary: Accept-Encoding
X-Powered-By: PHP/7.2.32
```

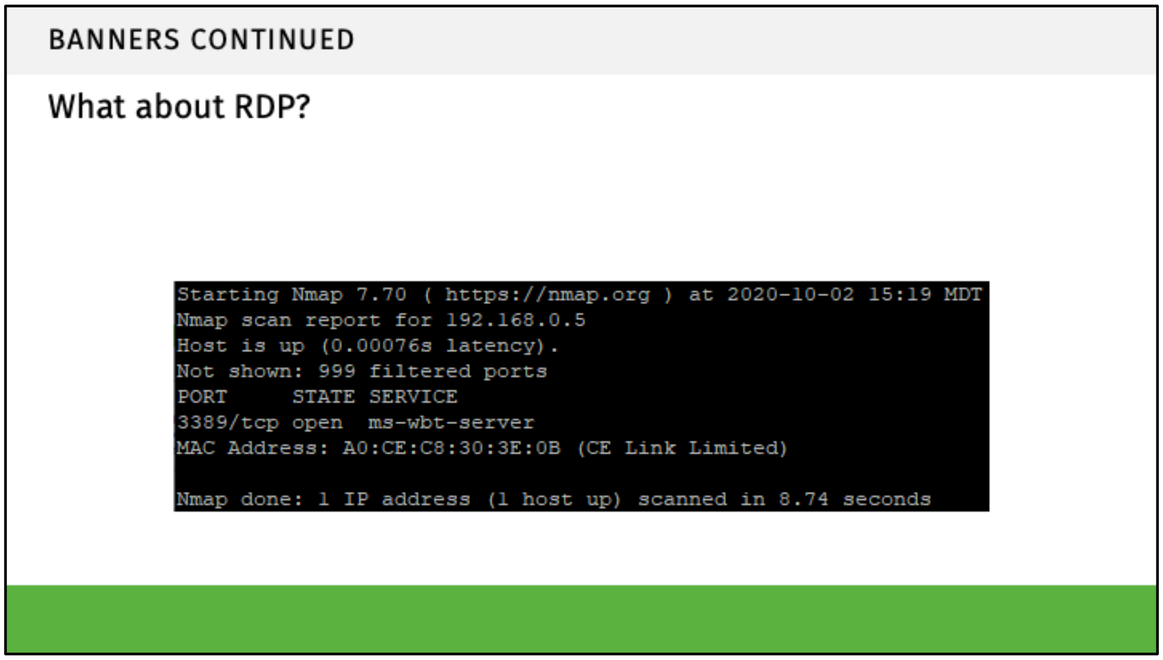
BANNERS CONTINUED

HTTP response headers **may** show the software version

What about SSH?

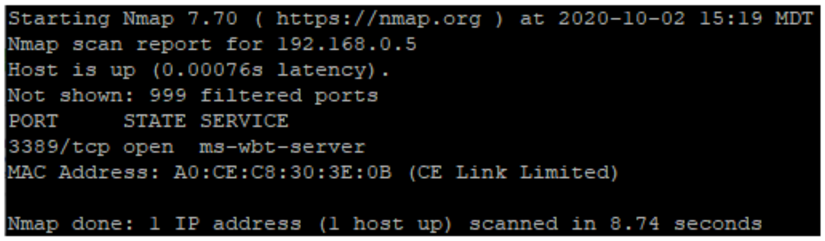


```
debug1: Connecting to 192.168.0.29 [192.168.0.29] port 22.
debug1: Connection established.
debug1: identity file /root/.ssh/id_rsa type -1
debug1: identity file /root/.ssh/id_rsa-cert type -1
debug1: identity file /root/.ssh/id_dsa type -1
debug1: identity file /root/.ssh/id_dsa-cert type -1
debug1: identity file /root/.ssh/id_ecdsa type -1
debug1: identity file /root/.ssh/id_ecdsa-cert type -1
debug1: identity file /root/.ssh/id_ed25519 type -1
debug1: identity file /root/.ssh/id_ed25519-cert type -1
debug1: identity file /root/.ssh/id_xmss type -1
debug1: identity file /root/.ssh/id_xmss-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_8.0
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.0
debug1: match: OpenSSH_8.0 pat OpenSSH* compat 0x04000000
```



BANNERS CONTINUED

What about RDP?



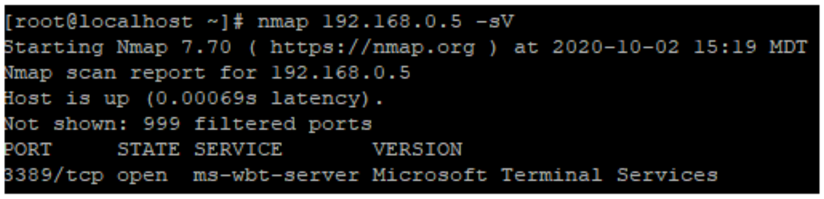
```
Starting Nmap 7.70 ( https://nmap.org ) at 2020-10-02 15:19 MDT
Nmap scan report for 192.168.0.5
Host is up (0.00076s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server
MAC Address: A0:CE:C8:30:3E:0B (CE Link Limited)

Nmap done: 1 IP address (1 host up) scanned in 8.74 seconds
```

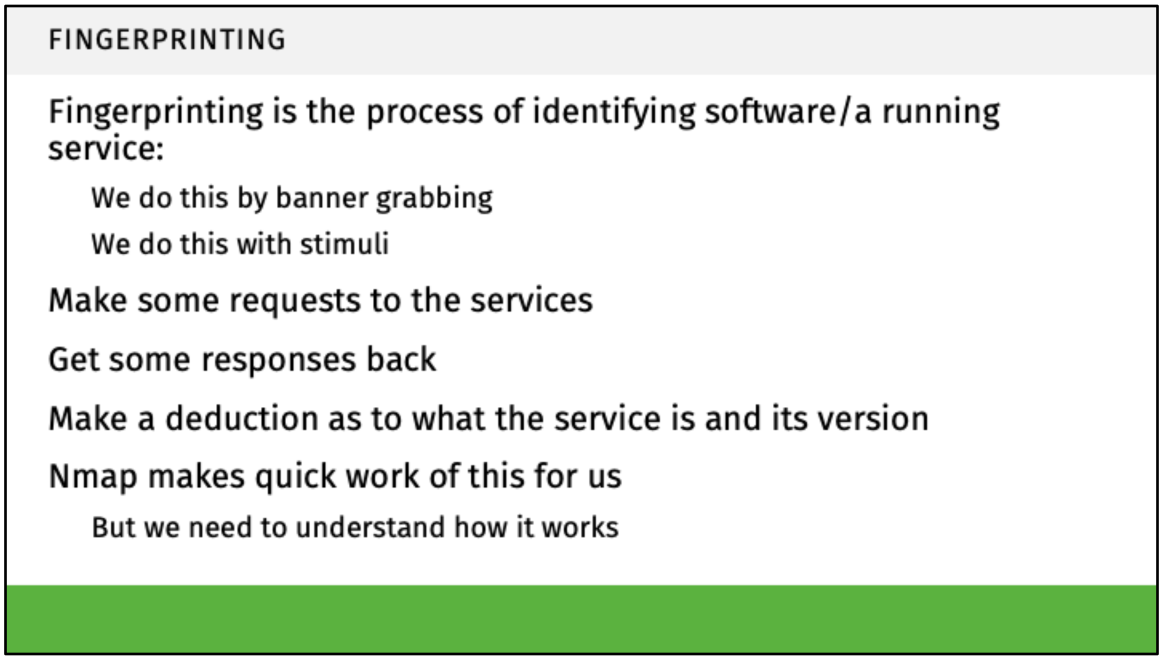
BANNERS CONTINUED

What about the version?

What if the version is not shown?



```
[root@localhost ~]# nmap 192.168.0.5 -sV
Starting Nmap 7.70 ( https://nmap.org ) at 2020-10-02 15:19 MDT
Nmap scan report for 192.168.0.5
Host is up (0.00069s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
```



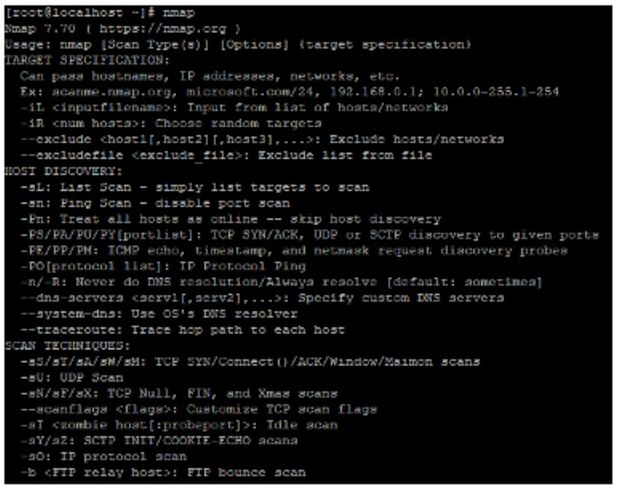
FINGERPRINTING

- Fingerprinting is the process of identifying software/a running service:
  - We do this by banner grabbing
  - We do this with stimuli
- Make some requests to the services
- Get some responses back
- Make a deduction as to what the service is and its version
- Nmap makes quick work of this for us
  - But we need to understand how it works

NMAP

Network mapping tool

Standard tool, high quality, used by professionals everywhere



```
[root@localhost ~]# nmap
Nmap 7.70 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
```