**Outline**

INTRODUCTION

ADRIAN

1. Module 1: Active Reconnaissance, Scanning, and Introduction to TCP/IP for pen-testers
2. Module 2: Basics of using nmap. Using nmap for host discovery
3. Module 3: Basics of using nmap. Understanding nmap output

JEREMY

1. Module 4: Using nmap for port and service discovery

LUNCH

1. Module 5: Using nmap timing options to aid performance
2. Module 6: Management reporting

MARTIN

1. Module 7: Using nmap scripting engine. Blurring the line between scanning and exploitation
2. Module 8: Metasploit Integration
3. Martin’s open-mike time
4. OPTIONAL: Module 9: Firewall Evasion

ANYONE, EVERYONE

1. OPTIONAL: Module 10: Other tools

**Introduction**

**Introduction**

**Module 1: Active Reconnaissance, Scanning, and Introduction to TCP/IP for pen-testers**

**What are the steps of a pen test and what are we covering?**

1. Passive Reconnaissance
2. Active Reconnaissance
3. Exploitation
4. Post Exploitation
5. Reporting

**What is active reconnaissance?**

1. Exposed Host Discovery
2. Exposed Port Discovery
3. Service Identification
4. Operating System Identification

**What are some active reconnaissance tools?**

* ping
* traceroute
* nmap
* zmap
* db\_nmap
* Metasploit auxiliary/scanning modules
* Amap
* others

**How does TCP/IP connection handshaking work?**

1. syn
2. syn-ack
3. ack
4. rst

**How do you know what is your network range?**

1. Local network
	1. Check gateway and subnet mask
2. Remote network
	1. Whois

**What is NMAP?**

1. Exposed Host Discovery
2. Exposed Port Discovery
3. Service Identification
4. Operating System Identification
5. Vulnerability Discovery

**Module 2: Basics of using nmap. Using nmap for host discovery**

**Basic Target Specification**

1. Hostname: nmap [scanme.nmap.org](http://www.nmap.org), www.irongeek.com
	1. Warning: Requires DNS resolution
2. IP Address
	1. nmap 10.0.0.0, 192.168.1.1
3. Networks
	1. nmap 10.0.0.0-255
	2. nmap 192.168.1.0/24
4. Options

 -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**

1. List targets (sanity check - do not scan): nmap -sL 10.0.0.0/24
2. Ping scan (local network): nmap -sn 10.0.0.0/24

ARP, Ethernet (len 6), IPv4 (len 4), **Request who-has 192.168.1.1 … tell 192.168.1.114**

ARP, Ethernet (len 6), IPv4 (len 4), **Reply 192.168.1.1 is-at 00:18:39:6e:e0:92**

1. Ping scan (remote network): nmap -sn scanme.nmap.org

**ARP local gateway**

ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.1 tell 192.168.1.114

ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.1 is-at 00:18:39:6e:e0:92

**Acquire IP of target via DNS**

IP (proto UDP (17)) 192.168.1.114.34058 > 8.8.8.8.53: A? scanme.nmap.org.

IP (proto UDP (17)) 8.8.8.8.53 > 192.168.1.114: A? scanme.nmap.org. 74.207.244.221

**Send ICMP echo request (standard ping)**

IP (proto ICMP (1)) 192.168.1 > 74.207.244.221: ICMP echo request

IP (proto ICMP (1)) 74.207.244 > 192.168.1: ICMP echo reply

**Send TCP SYN on port 443**

IP (proto TCP (6)) 192.168.1 > 74.207.244.221.443: Flags [S]

IP (proto TCP (6)) 74.207.244.221.443 > 192.168.1.114: Flags [R.]

**Send TCP ACK to port 80**

IP (proto TCP (6)) 192.168.1 > 74.207.244.221.80: Flags [.]

IP (proto TCP (6)) 74.207.244.221.80 > 192.168.1.114: Flags [R]

**Send ICMP time stamp query**

IP (proto ICMP (1)) 192.168.1 > 74.207.244.221: ICMP time stamp query

IP (proto ICMP (1)) 74.207.244 > 192.168.1: ICMP time stamp reply

1. Assume hosts online (skip discovery): nmap -Pn 10.0.0.0/24
2. Custom host discovery scan

-PS/PA/PU/PY[port list]: TCP SYN/ACK, UDP or SCTP discovery to given ports

-PE/PP/PM: ICMP echo, timestamp, and net mask request discovery probes

-PO[protocol list]: IP Protocol Ping

1. DNS resolution options

-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

1. Traceroute option

--traceroute: Trace hop path to each host

**Module 3: Basics of using nmap. Understanding nmap output.**

**What’s happening?**

 --reason: Display the reason a port is in a particular state

 --open: Only show open (or possibly open) ports

Verbosity: -v[v][v]

 Debugging: -d[d][d]

 During scan:

 Increase: v or d

 Decrease: Shift-v or Shift-d

**Module 4: Using nmap for port and service discovery**

**Controlling scan protocol**

**Lab**

NOTE: tcpdump -i eth1 -nn

Add (host <target IP>) to filter

 **What are the scan types?**

 **Reference**: <http://nmap.org/book/man-port-scanning-techniques.html>

-sS: SYN scan

Pros: Reliable. Works on any compliant network stack. Fast.

Cons: Requires root privileges

-sT: Connect scan

 Pros: Works without root privileges. Complements service detection.

 Cons: Ties up more scanner and target resources. Slower than SYN scan.

-sU: UDP scan

 Pros: Only scan-type that can locate UDP services

 Cons: Very slow when port closed

-sN; -sF; -sX, -sM (TCP NULL, FIN, and Xmas scans, Maimon)

 Pros: May get past packet filter when SYN blocked

Cons: Only detects if port closed or not (but cannot be sure port is open). Only works on some operating systems.

-sA: ACK scan

 Pros: Detects if firewall is stateful. Can map firewall rules for packet filters.

 Cons: Most firewalls are stateful.

--scanflags <flags>: Customize TCP scan flags

 Pros: Can choose any flags

 Cons: Requires advanced knowledge of TCP

-sO: IP protocol scan

 Pros: Can determine which protocols a host supports

 Cons: Not a port scan

**How to specify which ports to scan**

 **Lab**

 **What happens if service on port differs from service listed in nmap-services**

* Compare FTP service on Metasploitable-2
	+ **nmap 192.168.56.101 versus nmap -sV 192.168.56.101**

 **What happens if services file is modified? (Default versus modified file)**

* nmap --datadir $(pwd) 192.168.56.101
* nmap --datadir /usr/share/nmap/ 192.168.56.101

**What are top ports?**

**What is the format of the services (top ports) files?**

Default: scans “top 1000” ports

 Services file: /usr/share/nmap/nmap-services

 Reference: <http://nmap.org/book/nmap-services.html>

 Column 1: service name (output in SERVICE column of Nmap output)

 Column 2: port number/protocol

Column 3: port frequency

-F: Fast mode - Scan fewer ports than the default scan (top 100 ports)

--top-ports <number>: Scan <number> most common ports

--port-ratio <ratio>: Scan ports more common than <ratio>

-p <port ranges>: Only scan specified ports

 Examples:

 Scan port 22: -p22

 Scan all ports: -p1-65535

 -p1-

 -p-65535

 -p-

 Specify port and protocol: -p U:53,111,137,T:21-25,80

 -r: Scan ports consecutively - don't randomize

 Pros: Useful for testing what nmap is doing (i.e. demos)

 Cons: Obvious

**How to detect the operating system**

 **How does nmap identify operating systems?**

 Note: Operating system database

/usr/share/nmap/nmap-os-db

 **What happens if nmap tries to identify Kali?**

 **nmap-os-db format**

fingerprint blocks

* operating system's name
* general classification
* response data

**NMAP Options for OS identification**

-O: Enable O/S detection

--osscan-limit: Limit OS detection to promising targets (1 open/1 closed port)

--osscan-guess: Guess OS more aggressively

 **What if OS cannot be identified?**

 Note: Submission link - <http://nmap.org/submit/>

 Reference (Main): <http://nmap.org/book/nmap-os-db.html>

 Reference (Methods): http://nmap.org/book/osdetect-methods.html

**How to discover which services are running on each port**

 **How does nmap identify services?**

 **Lab**

tcpdump -i eth1 -nn

identify MySQL server (Metasploitable 2)

**What is the format of the files?**

**Reference**: Nmap service version file: /usr/share/nmap/nmap-service-probes

**How to customize the nmap services file to focus on certain services (i.e. web services)**

 **NMAP service version detection options**

-sV: Probe open ports to determine service/version info

--version-intensity <level>: Set from 0 (light) to 9 (try all probes)

--version-light: Limit to most likely probes (intensity 2)

--version-all: Try every single probe (intensity 9)

**How to use custom nmap-service-probes file**

**Example**: /root/nmap-workshop/modified-service-probes-file/nmap-service-probes

**Module 5: Using nmap timing options to aid performance**

**How do NMAP timing options balance scan performance with detection accuracy?**

**What are the options?**

--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes

--min-parallelism/max-parallelism <numprobes>: Probe parallelization

--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies probe round trip time.

--max-retries <tries>: Caps number of port scan probe retransmissions.

--host-timeout <time>: Give up on target after this long

--scan-delay/--max-scan-delay <time>: Adjust delay between probes

--min-rate <number>: Send packets no slower than <number> per second

--max-rate <number>: Send packets no faster than <number> per second

TIMING AND PERFORMANCE:

 Options which take <time> are in seconds, or append 'ms' (milliseconds),

 's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).

**What option templates are available?**

-T<0-5>: Set timing template (higher is faster)

Speed: T1

--------------- Timing report ---------------

 hostgroups: min 1, max 100000

 rtt-timeouts: init 15000, min 100, max 15000

 max-scan-delay: TCP 1000, UDP 1000, SCTP 1000

 parallelism: min 0, max 1

 max-retries: 10, host-timeout: 0

 min-rate: 0, max-rate: 0

---------------------------------------------

Speed: T2

--------------- Timing report ---------------

 hostgroups: min 1, max 100000

 rtt-timeouts: init 1000, min 100, max 10000

 max-scan-delay: TCP 1000, UDP 1000, SCTP 1000

 parallelism: min 0, max 1

 max-retries: 10, host-timeout: 0

 min-rate: 0, max-rate: 0

---------------------------------------------

Speed: T3

--------------- Timing report ---------------

 hostgroups: min 1, max 100000

 rtt-timeouts: init 1000, min 100, max 10000

 max-scan-delay: TCP 1000, UDP 1000, SCTP 1000

 parallelism: min 0, max 0

 max-retries: 10, host-timeout: 0

 min-rate: 0, max-rate: 0

---------------------------------------------

Speed: T4

--------------- Timing report ---------------

 hostgroups: min 1, max 100000

 rtt-timeouts: init 500, min 100, max 1250

 max-scan-delay: TCP 10, UDP 1000, SCTP 10

 parallelism: min 0, max 0

 max-retries: 6, host-timeout: 0

 min-rate: 0, max-rate: 0

---------------------------------------------

Speed: T5

--------------- Timing report ---------------

 hostgroups: min 1, max 100000

 rtt-timeouts: init 250, min 50, max 300

 max-scan-delay: TCP 5, UDP 1000, SCTP 5

 parallelism: min 0, max 0

 max-retries: 2, host-timeout: 900000

 min-rate: 0, max-rate: 0

---------------------------------------------

**How does accuracy compare accuracy and elapsed time at different speeds?**

**Lab** Start scan with -ddd option to show timing configuration

* nmap -vvv -ddd -T1 192.168.56.101

**Module 6: Management reporting**

**How to create reports to later phases of pen-testing and management**

**What are the options**

OUTPUT:

 -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,

 and Grepable format, respectively, to the given filename.

 -oA <basename>: Output in the three major formats at once

 --packet-trace: Show all packets sent and received

 --iflist: Print host interfaces and routes (for debugging)

 --log-errors: Log errors/warnings to the normal-format output file

 --append-output: Append to rather than clobber specified output files

 --resume <filename>: Resume an aborted scan

 --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML

 --webxml: Reference stylesheet from Nmap.Org for more portable XML

 --no-stylesheet: Prevent associating of XSL stylesheet w/XML output

1. Creating and using the grepable format
	1. nmap -oG output.gnmap 10.0.0.0
2. Creating and using the XML format
	1. Importing to other tools (Martin will cover this)
	2. Setting the stylesheet to local directory
	3. Locating and copying nmap.xsl to local directory
		1. cp $(nmap.xsl) .
3. Create all three outputs at once
4. Create XML that can be imported (This will be shown in next module)
5. How does XSL works
6. How does XML know where XSL can be found
7. How to fix missing XSL
8. How to customize the XSL

**Module 7: Using nmap scripting engine. Blurring the line between scanning and exploitation**

**The NMAP Scanning Engine (NSE)**

1. How to specify which scripts to run.
2. How to use wildcards
3. How scripts work
4. Overview of the scripts available
5. Demonstration of select scripts
6. Discuss some scripts helpful in your job

 -sC: equivalent to --script=default

 --script=<Lua scripts>: <Lua scripts> is a comma separated list of directories, script-files or script-categories

 --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts

 --script-args-file=filename: provide NSE script args in a file

 --script-trace: Show all data sent and received

 --script-updatedb: Update the script database.

 --script-help=<Lua scripts>: Show help about scripts.

 <Lua scripts> is a comma separated list of script-files or

 script-categories.

**Module 8: Metasploit Integration**

1. Relationship/integration between nmap and metasploit
2. Plus the aux scanners built into metasploit
3. For each topic, show different interfaces (i.e. msfconsole, Metasploit "gopro" web GUI, Armitage, etc.).

Some topics

1. Msfconsole interface
2. Armitage interface
3. Metasploit web GUI “go pro” interface
4. How the tables are loaded (hosts, services, etc.)
5. how to use/reference/output those tables
6. how to load a previous nmap.xml into Metasploit (msfconsole and GUI)
7. db\_nmap via the Metasploit msfconsole
8. the nmap scan from the Metasploit Web GUI
9. the msfconsole auxiliary scanners
10. mapping the results to the exploit that should be used
11. other interesting topics related to active-discovery phase of pen testing

**Module 9: Firewall Evasion (NOTE: Option: Martin may have custom material)**

FIREWALL/IDS EVASION AND SPOOFING:

 -f; --mtu <val>: fragment packets (optionally w/given MTU)

 -D <decoy1, decoy2[,ME],...>: Cloak a scan with decoys

 -S <IP\_Address>: Spoof source address

 -e <iface>: Use specified interface

 -g/--source-port <portnum>: Use given port number

 --data-length <num>: Append random data to sent packets

 --ip-options <options>: Send packets with specified ip options

 --ttl <val>: Set IP time-to-live field

 --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address

 --badsum: Send packets with a bogus TCP/UDP/SCTP checksum

**Module 10: Other tools (NOTE: Optional - Time allowing, Adrian & Martin may have optional material)**

1. zmap
2. scan.io - <https://scans.io/>
3. nmap on Raspberry Pi or other alternative hardware