



# PREPARING A NETWORK ATTACK: "SCANNING"

Corso di Laurea Magistrale in Ingegneria Informatica

Prof. Simon Pietro Romano

spromano@unina.it

# HOW TO PREPARE A NETWORK ATTACK?

- Vital concepts for anyone looking to prepare for a network attack:
  - Footprinting:
    - The art of gathering information on the network. Also known as "network reconnaissance"
  - Scanning:
    - A detailed inspection of the attack perimeter, searching for potential entry points
  - Enumeration:
    - Probing identified services to uncover potential vulnerabilities

# SCANNING vs FOOTPRINTING

- Footprinting:
  - studying the environment to gather a wide range of information
- Scanning:
  - thoroughly surveying the attack perimeter to identify potential access points to the target organization's systems

# SCANNING: GOALS

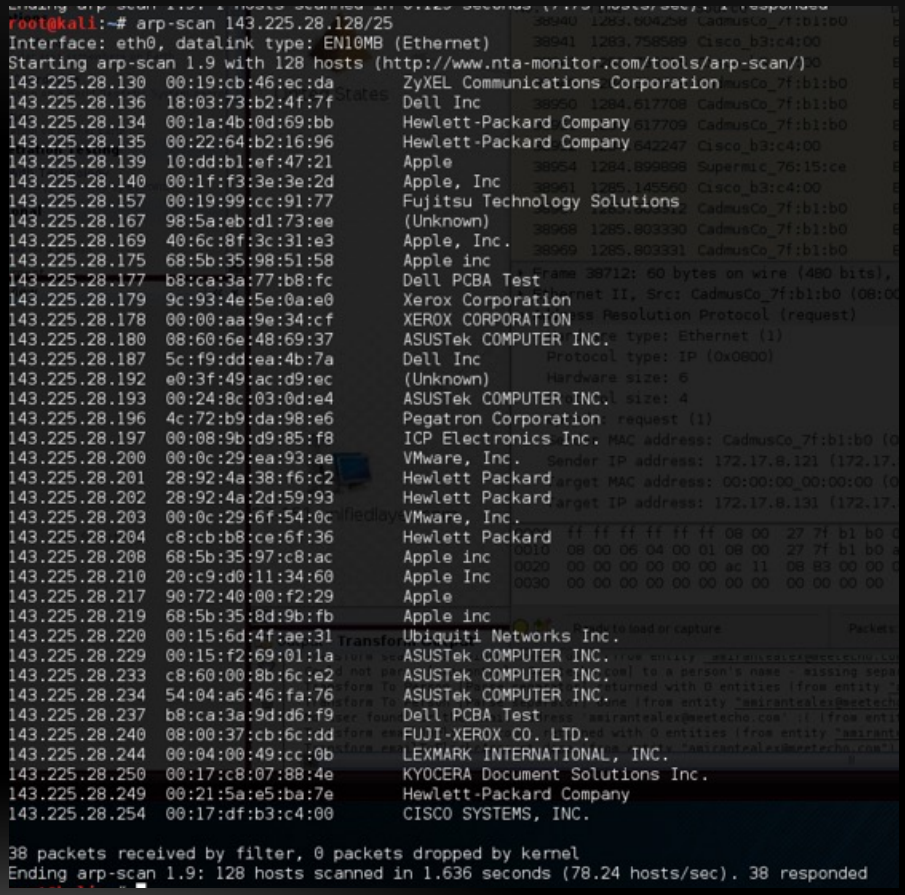
- Verifying which of the systems detected during footprinting are "alive", meaning they are listening for incoming traffic
- Circumventing any firewalls to perform reconnaissance on systems protected by filtering rules
- Maintaining anonymity (and minimizing intrusiveness) by using so-called "passive scanning" techniques

# DETECTING "ALIVE" SYSTEMS

- Basic technique: ping sweep
  - sending specific traffic to a target host and then analyzing the results
- Note:
  - The term "ping" historically refers to sending ICMP (Internet Control Message Protocol) Echo Request messages followed by receiving ICMP Echo Reply messages
  - Today, it has evolved to generally indicate sending messages belonging to various protocols, including ICMP, ARP (Address Resolution Protocol), TCP (Transmission Control Protocol), and UDP (User Datagram Protocol)

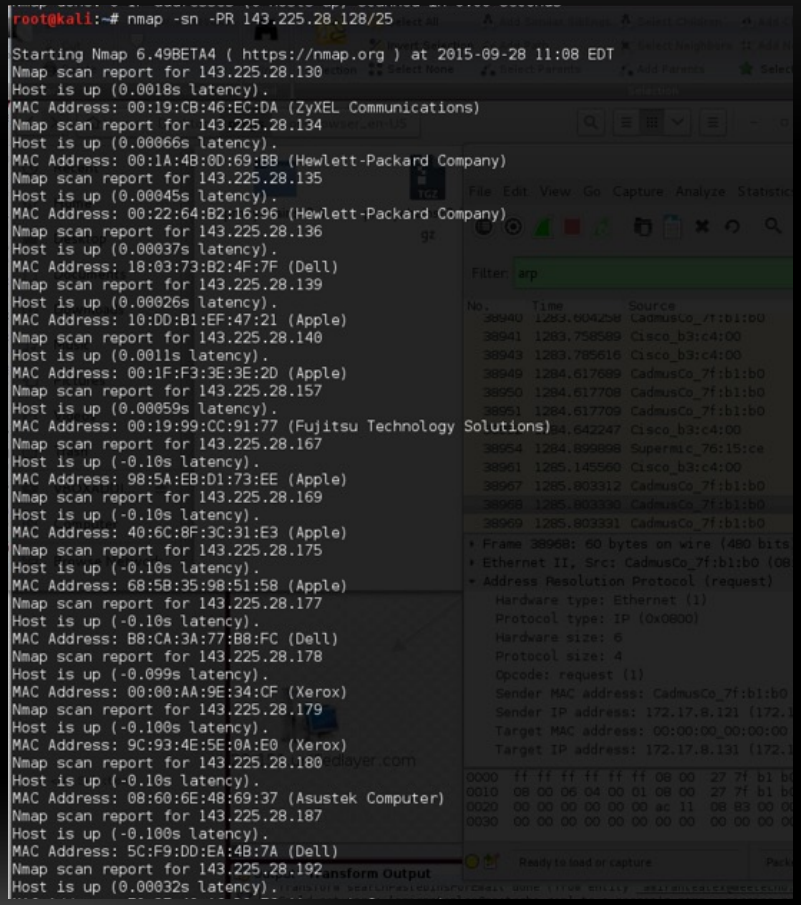
# arp-scan: DISCOVERING HOSTS ON A LOCAL NETWORK

- Useful when the attacker is on the same local network as the target
- It provides information about active hosts, including:
  - IP address
  - MAC address
  - Manufacturer of the local network adapter



# ARP SCANNING WITH "*nmap*"

- nmap: Netwok Mapper
  - a very powerful program for discovering network topology
  - capable of "mapping" network nodes and their services
  - supports ARP protocol scanning through the "-PR" option
  - limited to host discovery only using the "-sn" option
  - ...in reality, it can do much more than this (see next slides)



# DISCOVERY OF REMOTE HOSTS

- ARP works ONLY on the local network
- In the case of remote hosts, higher-level protocols are used:
  - ICMP
  - TCP/UDP
- Even in the case of remote hosts, there is no shortage of available tools...

# ICMP HOST DISCOVERY

- The most classic utility program: ping
  - sending an ICMP Echo request message...
  - ...receiving an ICMP Echo Reply message

```
root@kali:~# ping -c 4 143.225.28.167
PING 143.225.28.167 (143.225.28.167) 56(84) bytes of data.
64 bytes from 143.225.28.167: icmp_seq=1 ttl=64 time=0.245 ms
64 bytes from 143.225.28.167: icmp_seq=2 ttl=64 time=0.171 ms
64 bytes from 143.225.28.167: icmp_seq=3 ttl=64 time=0.198 ms
64 bytes from 143.225.28.167: icmp_seq=4 ttl=64 time=0.192 ms

--- 143.225.28.167 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.171/0.201/0.245/0.030 ms
```

# AGAIN WITH *nmap…*

- Options to use:
  - ***"-sn"*** → "no port scan"
  - ***"-PE"*** → send an ICMP Echo Request message
  - ***"--send-ip"*** → don't send ARP packets
- Note: in the absence of these options (and when executed as the "root" user), nmap would also do the following things:
  - ARP ping, send an ICMP Timestamp request, TCP pinging on ports 80 and 443!

```
root@kali:~# nmap -sn -PE --send-ip 143.225.28.167

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 13:39 EDT
Nmap scan report for 143.225.28.167
Host is up (0.00015s latency).
MAC Address: 98:5A:EB:D1:73:EE (Apple)
Nmap done: 1 IP address (1 host up) scanned in 13.21 seconds
```

# *nping:* HACKER'S PING PROGRAM!

- It allows for spoofing of the source MAC address, source IP address, and any other field in the packet
- It can be configured to send specific ICMP messages (e.g., "Timestamp" requests)

```
root@kali:~# nping -c 4 --icmp --icmp-type time 143.225.28.254

Starting Nping 0.6.49BETA4 ( http://nmap.org/nping ) at 2015-09-28 13:54 EDT
SENT (0.0022s) ICMP [143.225.28.168 > 143.225.28.254 Timestamp request (type=13/code=0) id=32993 seq=1 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=40 ]
RCVD (0.1983s) ICMP [143.225.28.254 > 143.225.28.168 Timestamp reply (type=14/code=0) id=32993 seq=1 orig=0 recv=64525994 trans=64525994] IP [ttl=255 id=18556 iplen=40 ]
SENT (1.0028s) ICMP [143.225.28.168 > 143.225.28.254 Timestamp request (type=13/code=0) id=32993 seq=2 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=40 ]
RCVD (1.2031s) ICMP [143.225.28.254 > 143.225.28.168 Timestamp reply (type=14/code=0) id=32993 seq=2 orig=0 recv=64526995 trans=64526995] IP [ttl=255 id=18556 iplen=40 ]
SENT (2.0033s) ICMP [143.225.28.168 > 143.225.28.254 Timestamp request (type=13/code=0) id=32993 seq=3 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=40 ]
RCVD (2.2025s) ICMP [143.225.28.254 > 143.225.28.168 Timestamp reply (type=14/code=0) id=32993 seq=3 orig=0 recv=64527995 trans=64527995] IP [ttl=255 id=18556 iplen=40 ]
SENT (3.0045s) ICMP [143.225.28.168 > 143.225.28.254 Timestamp request (type=13/code=0) id=32993 seq=4 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=40 ]
RCVD (3.2031s) ICMP [143.225.28.254 > 143.225.28.168 Timestamp reply (type=14/code=0) id=32993 seq=4 orig=0 recv=64528996 trans=64528996] IP [ttl=255 id=18556 iplen=40 ]

Max rtt: 200.074ms | Min rtt: 196.240ms | Avg rtt: 198.479ms
Raw packets sent: 4 (160B) | Rcvd: 4 (184B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 3.20 seconds
```

# TCP/UDP HOST DISCOVERY

- Useful in cases where the ICMP protocol is filtered for security reasons
  - A firewall protecting a web server might filter ICMP packets directed at it
  - …but would have to allow TCP segments directed at port 80
    - an attacker can thus perform probing by contacting TCP (on port 80) to determine if the host in question is "alive"
  - Of course, it is not easy to "guess" in advance which services are active on a host you want to know about:
    - blindly sending TCP segments addressed to variable port numbers on the target host
      - a time-consuming task and far from "silent"
        - the generated traffic is unlikely to escape a well-configured Intrusion Detection System!

# PORT PROBING WITH *nmap*

- Through the ***"-Pn"*** option, nmap probes 1000 ports of potential interest!

```
root@kali:~# nmap -Pn 143.225.28.169

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 14:09 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00077s latency).
Not shown: 969 closed ports, 26 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
443/tcp  open  https
1023/tcp open  netvenuechat
2049/tcp open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 97.63 seconds
```

# PROBING A SINGLE PORT

- A much more “scalable” solution
  - with *nmap:*
    - option to use:

    ***“-sS –p [port #] –open”***

```
root@kali:~# nmap -Pn -sS -p 22 --open 143.225.28.128/25

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 14:16 EDT
Nmap scan report for 143.225.28.139
Host is up (-0.076s latency).
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 10:DD:B1:EF:47:21 (Apple)

Nmap scan report for 143.225.28.187
Host is up (-0.077s latency).
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 5C:F9:DD:EA:4B:7A (Dell)
```

- Can also be implemented with tools such as *nping* or *SuperScan*

```
Nmap scan report for 143.225.28.220
Host is up (-0.076s latency).
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:15:6D:4F:AE:31 (Ubiquiti Networks)

Nmap scan report for 143.225.28.254
Host is up (-0.070s latency).
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:17:DF:B3:C4:00 (Cisco Systems)

Nmap done: 128 IP addresses (37 hosts up) scanned in 42.32 seconds
```

# PING SWEEP: DETECTION

- In a network:
  - the use of network-based Intrusion Detection systems is common
    - Snort (www.snort.org) is an example of a widely used open source network-based Intrusion Detection System
- On individual hosts:
  - tools for detecting and logging suspicious activities directed at the node are employed
    - examples of such tools include:
      - *scanlogd*, *ippl*, *Protolog, Filebeat* + *Logstash, …*

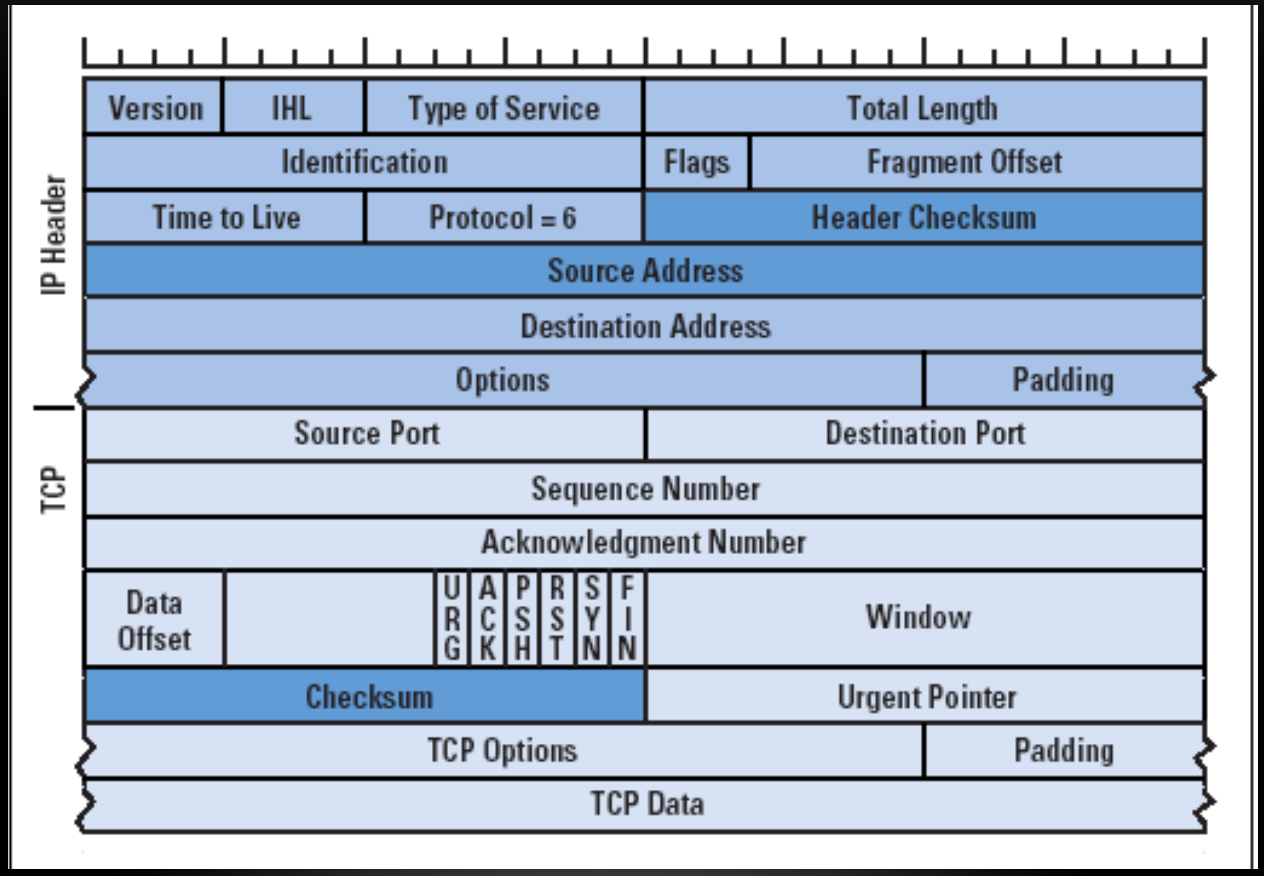
# PING SWEEP: PREVENTION

- Filtering the allowed ICMP message types:
  - e.g., Echo Request, Echo Reply, Host Unreachable and Time Exceeded only; and only if addressed to (a set of) hosts in the DMZ ("DeMilitarized Zone")
  - using Access Control Lists (ACLs) to allow ICMP traffic only towards a predefined set of external nodes
- Using ICMP as an end-user application ("userland daemon")
  - e.g.: *pingd* → deals with ICMP messages at the host level (outside of kernel)
- Beware!
  - ICMP, on a "compromised" host, might become a "back door" at Operating System's level:
    - "covert channel" for tunneling generic data inside ICMP packets:
      - e.g., *loki2* → http://phrack.org/issues/51/6.html

# AT THE DISCOVERY OF ACTIVE SERVICES

- Port scanning
  - probing network nodes to determine running or listening services
  - typically accomplished by sending packets to commonly used TCP and UDP ports
  - crucial for identifying potential vulnerabilities of a network node
  - also useful for determining other types of information such as:
    - the type and version of the operating system
    - running applications...

# LET'S REFRESH OUR MEMORY…



# TYPES OF SCANNING (1/3)

- TCP connect scan:
  - It completes the entire 3-way handshake (SYN, SYN+ACK, ACK)
  - Useful when scanning needs to be done as a non-privileged user
  - Slower and more easily traceable compared to other methods
- TCP SYN scan:
  - It doesn't complete the handshake
    - it sends SYN and waits for SYN+ACK without sending the final ACK
  - Useful for determining the presence of "listening" services
    - the service is absent (on that port) if an RST + ACK is received from the remote node
  - More robust ("stealthy") than the previous method
  - Often not captured in the target's system logs
  - Note: It can potentially cause a Denial of Service condition on the target
    - high number of "half-open" connections!

# TYPES OF SCANNING (2/3)

- TCP FIN scan:
  - Sends a FIN segment to a specific port on the target host
  - RFC 793 → respond with an RST segment if the port is closed, ignore the segment otherwise
- TCP Xmas Tree scan:
  - Sends a segment with the SYN, FIN, URG, and PUSH flags all set
  - RFC 793 → respond with an RST segment if the port is closed, ignore the packet otherwise
- TCP Null scan:
  - Sends a TCP segment with all flags set to 0
  - RFC 793 → respond with an RST segment if the port is closed, ignore the packet otherwise

Note: Typically, these three techniques work only for Unix-based TCP/IP stacks

- TCP ACK scan:
  - Sends a TCP segment with the ACK flag set
  - Useful for identifying firewalls that implement simple filtering rules
    - data from established connections (ACK = 1) are allowed without further inspection of the incoming packet

# TYPES OF SCANNING (3/3)

- TCP Window scan:
  - only useful for systems like FreeBSD or AIX
  - exploits an anomaly in reporting the window size
    - sends an ACK and expects to receive an RST:
      - RST with WIN = 0 → port closed
      - RST with WIN > 0 → port open!
- TCP RPC scan:
  - specific to UNIX systems
  - detects and identifies services of the Remote Procedure Call (RPC) type
- UDP scan:
  - sends a UDP packet to a specific port on the target:
    - if it receives an "ICMP Port Unreachable" → port closed.
  - much less reliable than TCP-based techniques due to the high likelihood of UDP traffic being filtered by the destination network/host

# PORT SCANNING WITH *nmap*

- TCP SYN scan…

```
root@kali:~# nmap -sS 143.225.28.169

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 05:45 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00073s latency).
Not shown: 929 closed ports, 66 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
443/tcp  open  https
1023/tcp open  netvenuechat
2049/tcp open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 94.74 seconds
```

- TCP SYN scan with output logged in an XML file

```
root@kali:~# nmap -sF 143.225.28.169 -oX pippozzo.xml

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 05:56 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00013s latency).
All 1000 scanned ports on 143.225.28.169 are open|filtered
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 21.54 seconds
```

# SCANNING WITH 'DECOY'…

- Usage of the so-called "decoy" option:

> The -D option allows you to specify Decoys. This option makes it look like those decoys are scanning the target network. It does not hide your own IP, but it makes your IP one of a torrent of others supposedly scanning the victim at the same time. This not only makes the scan look more scary, but reduces the chance of you being traced from your scan (difficult to tell which system is the "real" source).

- Requires the "spoofing" of a valid IP address to avoid flooding the target system with SYN segments
- With this type of scan, it is difficult for the target system to identify the real initiator of the scan (its data gets mixed with the data of the decoy node)

Source of the scan

```
root@kali:~# nmap -sS 143.225.28.167 -D 143.225.28.169

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 09:46 EDT
Nmap scan report for 143.225.28.167
Host is up (0.00023s latency).
All 1000 scanned ports on 143.225.28.167 are closed
MAC Address: 98:5A:EB:D1:73:EE (Apple)

Nmap done: 1 IP address (1 host up) scanned in 69.42 seconds
```

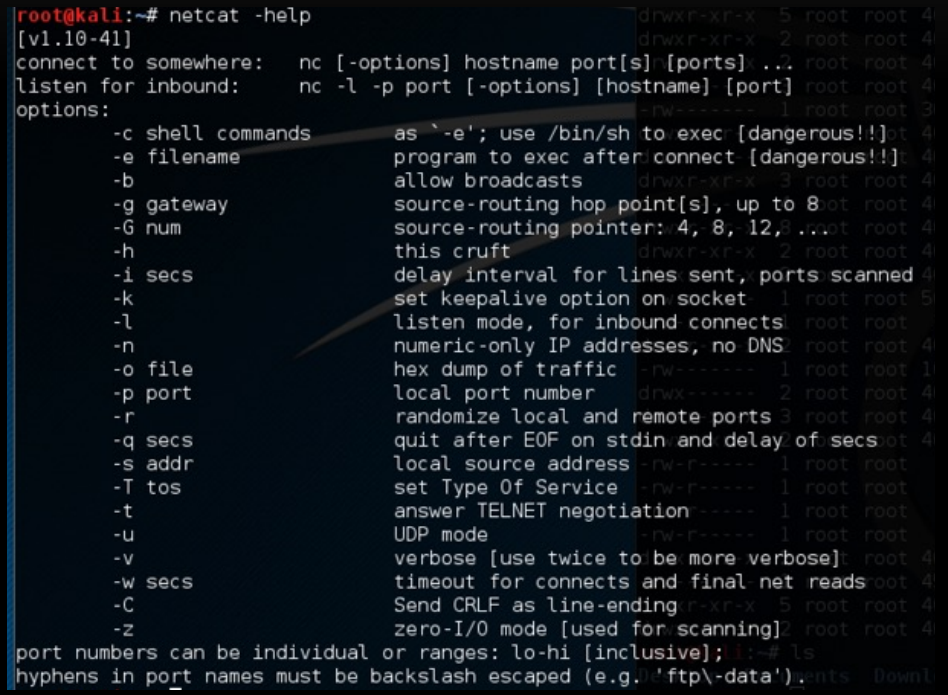
Decoy Host

Filter: ip.src==143.225.28.167 — Expression… Clear Apply Save

| Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 6565→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 100→36607 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 50001→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 3390→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5102→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 16993→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5102→36607 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5679→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5999→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5405→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 3001→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 5405→36607 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 143.225.28.167 | 143.225.28.169 | TCP | 60 | 4005→36606 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

# ONE MORE TOOL: *netcat*

- The "Swiss Army knife" of security
- Useful when you want to minimize your traces in a compromised system
- Scanning based on TCP or UDP (option "-u")
- Many configuration options...



# PORT SCANNING: COUNTERMEASURES

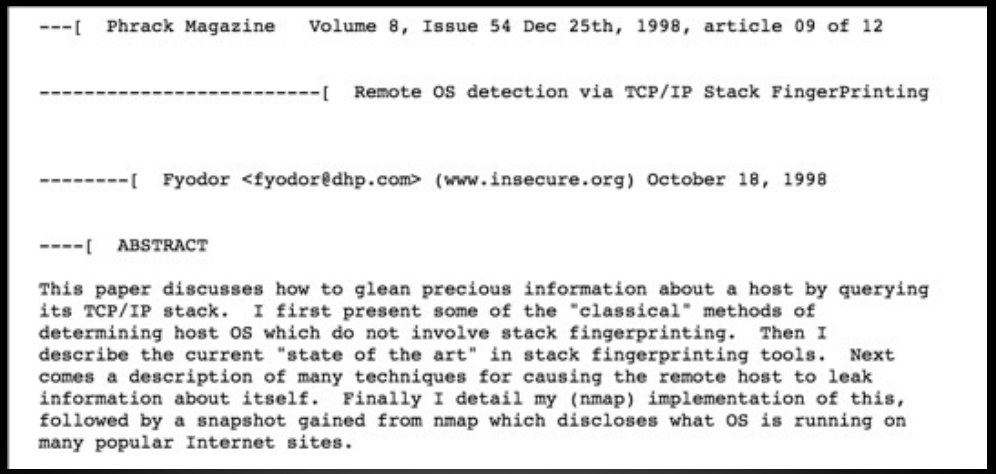
- As usual:
  - Detection:
    - Use a Network-based Intrusion Detection System like Snort (www.snort.org)
    - Use log file analysis tools, e.g., scanlogd
    - Be aware that the IP addresses of supposed attackers are typically cleverly forged (spoofing)...
      - ...taking countermeasures against such addresses almost always means dealing with the wrong "person"!
  - Prevention:
    - There's really not much to do here (it's difficult to convince a hacker of the futility of scanning our network nodes)
    - The only useful advice: disable ALL services considered unnecessary!

# DISCOVERY OF THE OPERATING SYSTEM TYPE

- From the simplest techniques, such as "banner-grabbing"
  - see the lesson on footprinting
- …to more advanced methods:
  - the so-called "stack fingerprinting":
    - searching for clear signs of a specific type of operating system based on the services available on a node (e.g., port scanning)
    - Examples:
      - active ports 135 (Endpoint Mapper), 139 (NetBIOS), and 445 (Active Directory) → Windows!
      - active ports 22 (ssh), 111 (SUN RPC), and 2049 (NFS) → a high probability of a Unix-based system
  - This technique is based on the use of both active and passive methods

# ACTIVE STACK FINGERPRINTING

- A set of techniques aimed at quickly and reliably determining the type of operating system (OS) running on a network node
- This involves using detailed knowledge about how various OS manufacturers implement the Internet standard TCP/IP stack (RFC)
- Reliable estimations typically require relying on at least one open port on the node



```
---[  Phrack Magazine   Volume 8, Issue 54 Dec 25th, 1998, article 09 of 12

-------------------------[  Remote OS detection via TCP/IP Stack FingerPrinting


--------[  Fyodor <fyodor@dhp.com> (www.insecure.org) October 18, 1998


----[  ABSTRACT

This paper discusses how to glean precious information about a host by querying
its TCP/IP stack.  I first present some of the "classical" methods of
determining host OS which do not involve stack fingerprinting.  Then I
describe the current "state of the art" in stack fingerprinting tools.  Next
comes a description of many techniques for causing the remote host to leak
information about itself.  Finally I detail my (nmap) implementation of this,
followed by a snapshot gained from nmap which discloses what OS is running on
many popular Internet sites.
```

# PROBES FOR STACK FINGERPRINTING (1/2)

- FIN probe:
  - send a TCP FIN segment to an open port:
    - RFC793 → DO NOT answer!
    - Some Windows implementations (7, 200X, Vista) → Send FIN+ACK ☹
- Bogus flag probe:
  - Set an undefined flag within the header of a TCP SYN segment:
    - Linux → copies back the flag in question within the response segment (SYN+ACK) ☹
- "Don't Fragment bit" monitoring:
  - some OSs set such a bit (inside the IP header) by default, to increase performance
- TCP initial window size:
  - some TCP implementations associate a constant value with the receiver window
- ACK value:
  - RFC793 → "Sequence # + 1"
  - Some TCP implementations → "Sequence #" ☹

# PROBES FOR STACK FINGERPRINTING (2/2)

- ICMP message quoting:
  - different OSs "copy back" into the response different parts of the original message when crafting an ICMP error message
- Type of Service (TOS):
  - analysis  of the TOS field of the received ICMP “Port Unreachable” messages (should be se to ‘0’, but some OSs set this in a different way…)
- Fragmentation Handling:
  - different OSs implement in different ways the reconstruction of an IP datagram starting from "overlapping" fragments
- TCP Options:
  - RFC1323 → specifies NEW optons for TCP (“no operation”, “window scale”, etc)…
  - …this allows to tell newer stacks apart from older ones (compliant with just RFC793)

# OS DETECTION WITH *nmap*

- Use of the ***"-O"*** option, based on the adoption of most of the mentioned stack fingerptinting techniques

```
root@kali:/etc/snort/rules# nmap -O 143.225.28.169

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-30 02:42 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00081s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
443/tcp  open  https
1023/tcp open  netvenuechat
2049/tcp open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)
Device type: general purpose|media device|phone
Running: Apple Mac OS X 10.7.X|10.9.X|10.8.X, Apple iOS 4.X|5.X|6.X
OS CPE: cpe:/o:apple:mac_os_x:10.7 cpe:/o:apple:mac_os_x:10.9 cpe:/o:apple:mac_os_x:10.8 cpe:/o:apple:iphone_os:4
cpe:/a:apple:apple_tv:4 cpe:/o:apple:iphone_os:5 cpe:/o:apple:iphone_os:6
OS details: Apple Mac OS X 10.7.0 (Lion) - 10.10 (Yosemite) or iOS 4.1 - 8.1.2 (Darwin 10.0.0 - 14.0.0)
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 51.39 seconds
```

# OS DETECTION: COUNTERMEASURES

- Detection:
  - the "advice" regarding the detection of generic port scanning activity also applies in this case
    - scans with specific options configured in probe packets (e.g., SYN flag in TCP) are a good indicator of stack fingerprinting activity
- Prevention:
  - It is quite challenging!
    - Preventing stack fingerprinting activity requires modifying the behavior of the operating system (at the source code level or through configuration parameters)
  - The only effective form of prevention involves relying on robust firewalls and/or proxies to protect the network resources of your systems

# PASSIVE STACK FINGERPRINTING

- Techniques designed to make fingerprinting activity more robust in terms of evading detection by IDS
- These techniques avoid proactively sending probe packets to target nodes
- They rely solely on monitoring and analyzing network traffic
- They require that the attacker is:
  - positioned at a "central" point in the network
  - capable of "listening" to traffic on a port that allows packet capture (a "mirrored port")
- Some examples of projects and tools for passive fingerprinting:
  - The "honeynet" project → http://honeynet.org/
  - The "siphon" tool → passive port mapping, OS identification, and topology discovery

# PASSIVE SIGNATURES

- Identification of specific traffic characteristics:
  - Time To Live (TTL):
    - Different operating systems use different default TTL values
  - TCP Window Size:
    - Different operating systems use different default values for the initial window size announced by a TCP receiver
  - Don't Fragment (DF) Bit:
    - Some operating systems set this bit high by default, while others do not
- Analyzing the traffic and comparing the analysis results with specific "signatures" in a custom-built database allows for the identification of the operating system (OS) of the nodes that generated the monitored data

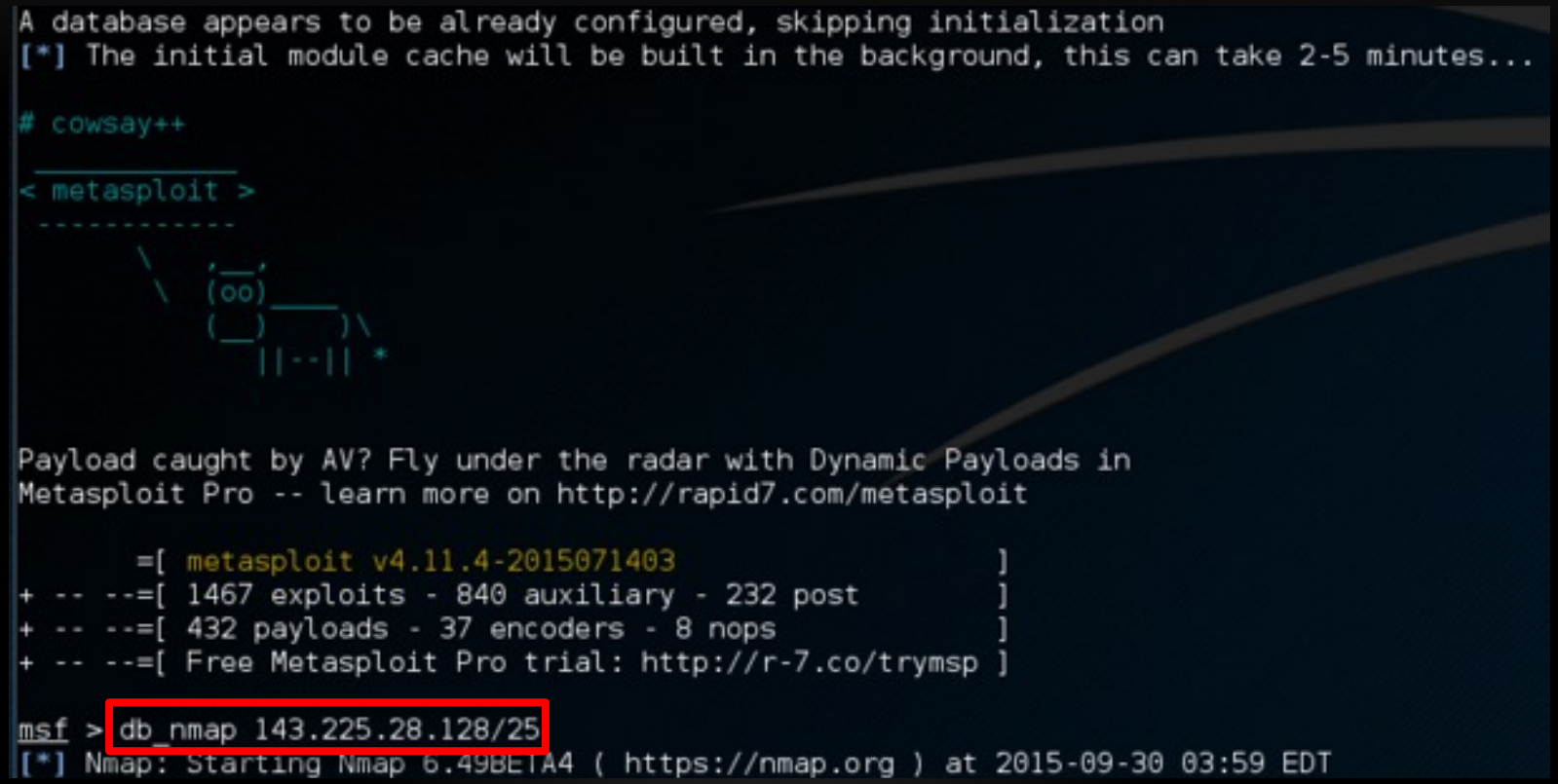
# STORING AND PROCESSING RESULTS

- Crucial activity for structurally analyzing all the information gathered during the scanning phases
- Typically achieved by importing the scan data into a dedicated database
- The scanning database becomes a valuable repository of data for:
  - extracting a comprehensive profile of the target system
  - identifying potential vulnerabilities through activities such as inference and correlation of information
  - preparing for subsequent attack phases by creating a structured plan of "offensive" actions

# MANAGING DATA WITH "*Metasploit*"

- A comprehensive platform for developing network attacks
- It includes an impressive collection of:
  - Fingerprinting tools
  - Attack payloads
  - Exploits for network systems
- The platform can import data into a database, allowing for specific queries to:
  - systematically study target systems
  - identify potential attack patterns

# THE Metasploit DATABASE



# Metasploit: ANALYZING DATA

When the scan is over…

```
[*] Nmap: 9103/tcp open  jetdirect
[*] Nmap: MAC Address: 00:17:C8:07:88:4E (Kyocera Document Solutions)
[*] Nmap: Nmap scan report for 143.225.28.254
[*] Nmap: Host is up (0.00072s latency).
[*] Nmap: Not shown: 998 closed ports
[*] Nmap: PORT   STATE SERVICE
[*] Nmap: 22/tcp open  ssh
[*] Nmap: 23/tcp open  telnet
[*] Nmap: MAC Address: 00:17:DF:B3:C4:00 (Cisco Systems)
[*] Nmap: Nmap scan report for 143.225.28.168
[*] Nmap: Host is up (0.0000010s latency).
[*] Nmap: All 1000 scanned ports on 143.225.28.168 are closed
[*] Nmap: Nmap done: 128 IP addresses (38 hosts up) scanned in 7023.90 seconds
msf >
```

```
msf > services

Services
========

host            port   proto  name            state     info
----            ----   -----  ----            -----     ----
143.225.28.134  21     tcp    ftp             open
143.225.28.134  80     tcp    http            open
143.225.28.134  631    tcp    ipp             open
143.225.28.134  7627   tcp    soap-http       open
143.225.28.134  14000  tcp    scotty-ft       open
143.225.28.134  280    tcp    http-mgmt       open
143.225.28.134  9100   tcp    jetdirect       open
143.225.28.134  443    tcp    https           open
143.225.28.134  515    tcp    printer         open
143.225.28.134  23     tcp    telnet          open
143.225.28.135  135    tcp    msrpc           open
143.225.28.135  139    tcp    netbios-ssn     open
143.225.28.135  21     tcp    ftp             open
143.225.28.135  3389   tcp    ms-wbt-server   open
143.225.28.135  1947   tcp    sentinelsrm     open
143.225.28.135  445    tcp    microsoft-ds    open
143.225.28.135  1723   tcp    pptp            open
143.225.28.136  3389   tcp    ms-wbt-server   open
```

…data is in the DB, ready for use!

# QUESTIONS?