NMAP – Network Scanning

Otacon22

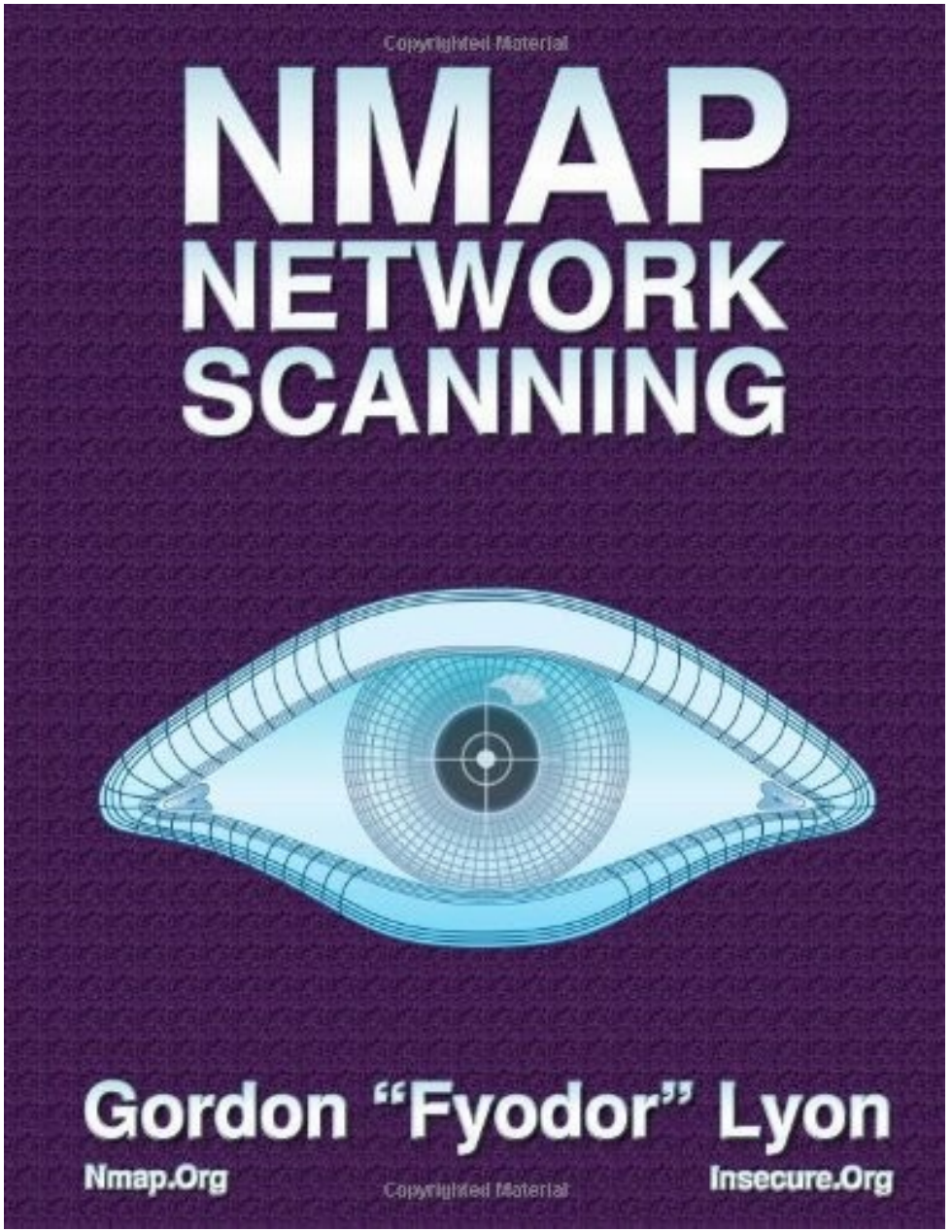
Who are you and what are you doing on my network?

- Studente ing.informatica
- GNU/Linux and Networking Enthusiast
- http://www.otacon22.it/
- http://www.github.com/Otacon22/
- *Network Address Translation Hater*™
- Radio ham callsign: IZ2YJE

Riferimenti: “IL” libro

Versione 4.76 (ora siamo alla 6.25)

Ma non è cambiato molto sulla struttura e le tecniche principali!



Riferimenti

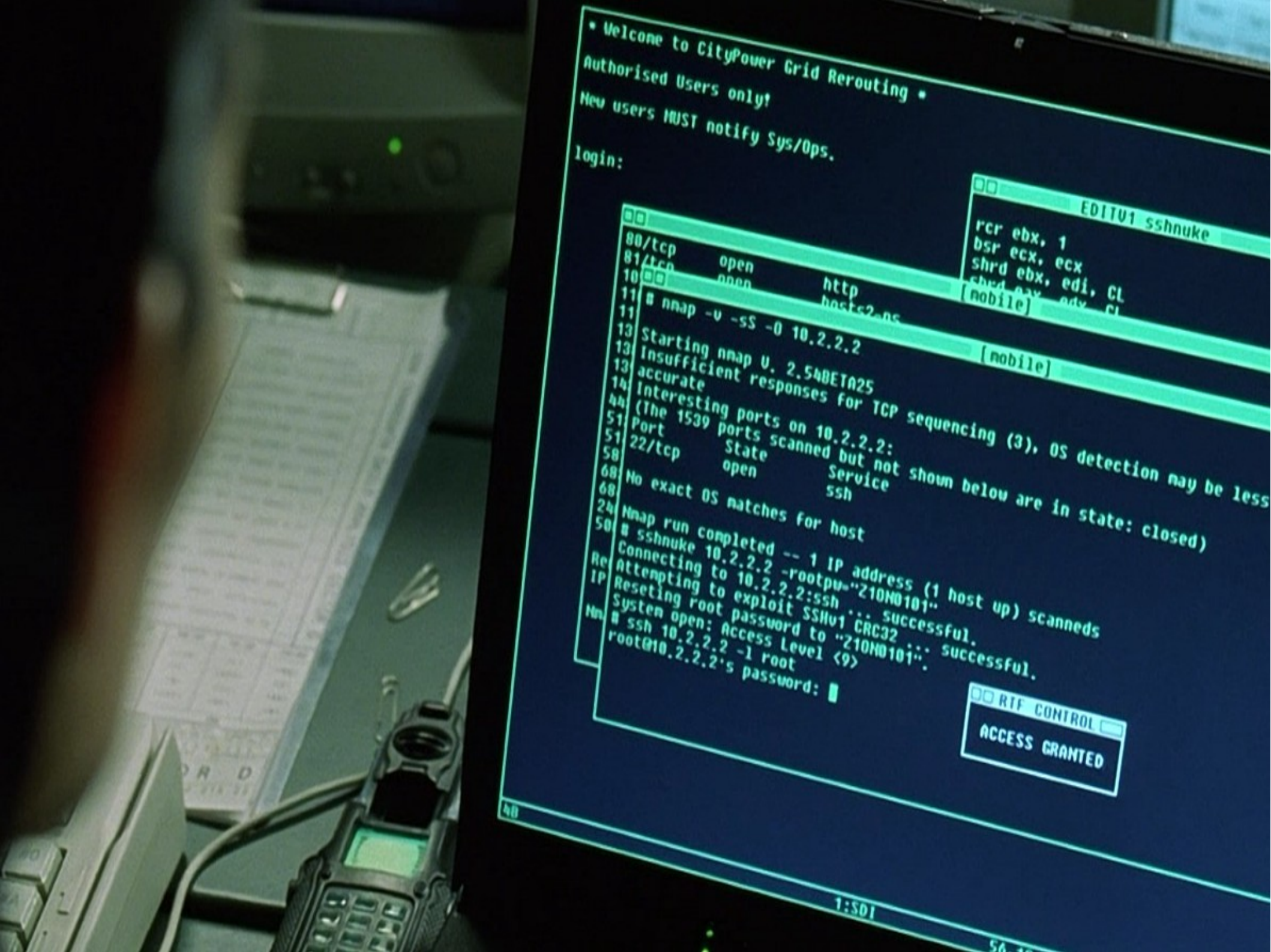
http://nmap.org/book/man.html

Hem...Esattamente parliamo di...?

- Nmap è un tool free ed open source per esplorazione e controlli di sicurezza su reti IP.
- È un tool che tenta di capire:
  - Quali host attivi ci sono sulla rete analizzata
  - Quali servizi sono offerti
  - Quali sistemi operativi sono in uso
  - Che tipo di firewall è in uso
  - ...
- Esiste anche un tool chiamato **Zenmap** che è l'equivalente di NMAP con interfaccia GUI.

Hem...Esattamente parliamo di...?

- Prima release nmap 1997
- Mi aspetto conoscenze base di reti TCP/IP
- Non affronterò tutte le funzionalità del tool ma buona parte (guardate l'altezza del libro che ho sul tavolo).
- Siete **caldamente** invitati a interrompermi durante il talk per domande


* Welcome to CityPower Grid Rerouting *
Authorised Users only!
New users MUST notify Sys/Ops.
login:
EDIT\1 sshnuke
rcr ebx, 1
bsr ecx, ecx
shrd ebx, edi, CL
80/tcp open http
[mobile]
nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may be less accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanned
sshnuke 10.2.2.2 -rootpw="Z1ON0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z1ON0101".
System open: Access Level <9>
ssh 10.2.2.2 -l root
root@10.2.2.2's password:
RIF CONTROL
ACCESS GRANTED
1:SD1

Le 9 fasi di uno scan Nmap

1. **Target enumeration**
2. **Host discovery (“ping” scanning)**
3. **Reverse-DNS resolution**
4. **Port scanning**
5. **Version detection**
6. **OS detection**
7. **Traceroute**
8. **Script scanning**
9. **Output**

1 Target enumeration

- In questa fase nmap “srotola” la lista di host su cui deve lavorare.
- Possiamo specificare gli host in notazione CIDR o simil-regexp se vogliamo fare matching su una singola rete
  - `nmap 172.20.69.0/24`
  - `nmap 172.20.69.* 192.168.10-42.*`
- Possiamo scegliere anche host random in una sottorete o sull'intera Internet
- Possiamo passare domini ovviamente
- Se usiamo nmap con l'opzione “-sL -n” viene fatta solo enumerazione degli host

2 Host discovery or “*ping scanning*”

- È inutile fare port scanning e altre operazioni su host che sono “down”: si **perde un sacco di tempo**
- Nel network scanning il tempo conta parecchio, specialmente se stiamo facendo scanning su reti remote!
- Quindi: si tenta di scoprire quali host sono attivi in rete.
- Si usa ICMP ping, ma non solo, ci sono tanti tipi di probe inviate (ARP se in rete locale, SYN su una porta TCP...)
- Poi si fa scanning solo sugli host “up”
- Si può comunque forzare lo scanning su tutto up/down
- Comodo per fare ping su intere sottoreti (no, non si può fare con broadcast ping, non funziona). In questo caso si può forzare **solo** host discovery con **-sP -n**

3 Reverse-DNS resolution

- Una volta scelti gli host su cui fare scanning, si effettua l'operazione di reverse-DNS lookup sugli host trovati “up”.
- Non avete idea della quantità di informazioni che si riesce a scoprire tramite i rDNS!
- Spesso ISP e aziende tengono rDNS ! (Comodo per rintracciare quale macchina sta facendo cosa)

- Questo step può essere tralasciato (ad es: sappiamo già l'host da analizzare, non ci interessano i reverse), utilizzando **-n**

- Oppure possiamo forzare a fare il reverse DNS anche degli host “down” con **-R**

4 Port scanning

- È l'operazione fondamentale di nmap: tentare di capire quali porte TCP/UDP/.. sono “aperte” e accettano pacchetti in ingresso su un host.
- Vengono inviate delle probe e le risposte (o non-risposte) permettono di capire in che stato è la porta (`open, closed, filtered` principalmente)
- Nmap dispone di **diversi** tipi di port scan e algoritmi che possono aumentare la velocità e accuratezza.
- Si può skippare il portscan con **-sP**

5 Version detection

- Potete dirmi: beh ma se trovo la porta 22 aperta non è detto che ci giri mica SSH dietro!
- Ecco, in nmap esiste la ***version detection*** che risolve proprio questo problema. Con essa nmap tenta di capire **il servizio che sta girando dietro una certa porta**
- Vengono inviate una serie di probe a livello applicativo, analizzate le risposte e confrontate con un database di signature note
- Disabilitato di default, si abilita con **-sV**

6 OS detection

- I diversi OS implementano gli standard e stack di rete in modo diverso.
- “Misurando” le differenze di reazione a certi “stimoli” è possibile capire lo stack TCP/IP che c'è dietro e quindi il sistema operativo.
- Vengono inviate una serie di probes di vari tipi (ICMP, TCP, UDP, …) e analizzate le risposte confrontando con un database di behaviours noti
- Di default disabilitato, si abilita con **-O**

7 Traceroute

- Avete mai provato traceroute su Linux e lamentati della sua lentezza? È dovuta a come è stato inventato
- Nmap contiene una implementazione alternativa di traceroute. Scoprite gli host sulla vostra route verso la destinazione in modo più veloce e completo.
- Vengono utilizzate un po' di “probe” di nmap verso i router intermedi (non solo ICMP come traceroute)
- Si abilita con **--traceroute**

8 Script scanning

- Nmap contiene l'**Nmap Scripting Engine** (**NSE**)
- Utilizza una serie di scripts speciali (ad-hoc per ogni servizio), per trovare informazioni aggiuntive su un servizio. Gli script sono in **LUA**, affiancati da una libreria nmap disegnata per reperire informazioni dalla rete.
- Esempio classico: script che lista le cartelle esposte su netbios/samba.
- Ce ne sono già un po' integrati. Spenti di default, Si abilitano con --**script** oppure **-sC**

9 Output

- Di default nmap vi stampa in output i risultati dello scanning, in un formato piuttosto comodo da leggere “a mano”
- Se l'output di nmap deve andare in pasto a grep/sed/awk potreste essere interessati ad avere un output strutturato “grepable” **-oG** <file>
- Output XML con **-oX** <file>

Grossi scan

- Le fasi sono queste, però nel caso in cui facciamo scan su grosse reti vengono ripetute tutte in più volte:
  - 1.Ping Scan di un sottogruppo della grossa rete
  - 2.Port Scan degli host up del sottogruppo
  - ...
  - GOTO 1
- Quindi si lavora su dei piccoli gruppi per volta.

Comando semplice

- Nmap è newbie-proof. Ha una manpage sconfinata ma anche un help da riga di comando abbastanza sintetico.
- Il comando: **nmap 10.12.45.1**
  - Lancia un rDNS sull'ip scelto
  - Avvia la prodedura di host discovery per controllare se è up
  - Se l'host è up parte la procedura di portscan
  - Vengono ritornati i risultati a schermo
  - **Tutto senza specificare nessuna opzione**

Analisi dei pacchetti

- Nmap vi aiuta a capire come funziona internamente:
- Con l'opzione **--packet-trace** si abilita una modalità verbose che vi mostra pacchetto per pacchetto quello che viene inviato a ricevuto da nmap
- Con l'opzione --**reason** a fianco ad ogni porta viene mostrato il motivo per cui è in quello stato (vedremo dopo)

```
:~$ sudo nmap -n -PE --packet-trace -sP 10.188.160.1

Starting Nmap 5.21 ( http://nmap.org ) at 2013-04-17 23:05 CEST
SENT (0.0920s) ICMP 192.168.0.229 > 10.188.160.1 echo request
(type=8/code=0) ttl=40 id=61131 iplen=28
RCVD (0.1010s) ICMP 10.188.160.1 > 192.168.0.229 echo reply
(type=0/code=0) ttl=126 id=61131 iplen=28
Nmap scan report for 10.188.160.1
Host is up (0.0093s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
```

Perché fare network scanning?

- Controllo sicurezza di grosse aziende
- Controllo macchine collegate e possibili vulnerabilità esposte dai dipendenti consapevolmente o inconsapevolmente
- Molte aziende hanno degli IDS, ma anche dei sistemi di portscanning automatici sulla rete aziendale!
- Qualcuno potrebbe chiedervi a spot di fare del penetration testing su una rete aziendale

Problemi legali del network scanning

- Se dovete fare pentesting la cosa migliore è avere una autorizzazione scritta prima di iniziare gli scan
- Gli ISP potrebbero accorgersi e farvi delle domande
- Open Source Security Testing Methodology Manual (OSSTMM – http://www.osstmm.org/)
- I network admin della rete che scannate potrebbero fregarsene, ma il vostro ISP potrebbe bloccarvi citando le regole per il servizio.
- Regola d'oro: **non rompete troppo le scatole**

Il network scanning “fa male” ?

- Nmap non manda pacchetti corrotti o disegnati per far crashare le macchine remote, manda gli header in base agli standard.
- Se la macchina remota crasha è un problema del vendor.
- Raramente qualcuno segnata di sistemi crashati con uno scan di nmap.
- Sapere che una macchina crasha con nmap può essere un vantaggio dal punto di vista della security preventiva!

Enumeration

- Abbiamo visto che gli host possono essere inseriti in notazione CIDR, possiamo anche inserire a partire da un file con una lista o escludere un gruppo di host

Enumeration

nmap scanme.nmap.org
nmap scanme.nmap.org/32
nmap 64.13.134.52

nmap scanme.nmap.org/24
nmap 64.13.134.52/24
nmap 64.13.134.0-255

nmap 64.13.134.52/24 –exclude scanme.nmap.org,insicure.org
nmap 10.0.0.8/8 –exclude 10.6.0.0/16,ultra-sensitive-host.company.com

Egrep '^lease' /var/lib/dhcp/dhcpd.leases | awk '{print $2}' | nmap -iL -

nmap -6 2001:800:40:2a03::3

Trovare gli IP da analizzare

- Partire dai DNS dell'azienda/organizzazione da analizzare e iniziare a controllare i vari tipi di record: NS,MX,A,AAA, …
- Controllare se qualche mona ha lasciato AXFR type acceso pubblico (zone transfer)
- Se parliamo di una organizzazione con un Autonomous System:
  - Whois su un ip dell'organizzazione, trovo AS number
  - Whois su AS number, trovo tutti i prefissi
  - Oppure controllo route BGP (http://bgp.he.net/)

Setacciare gli IP da analizzare

- Una volta scoperto il prefisso di rete che ci interessa magari non vogliamo fare scanning di tutto (vedi esempio oɔıuɔǝʇıʃod).
- Ancora rDNS: è facile trovare host chiamati tipo:
  - Firewall.example.com
  - Access-point.example.com
- Prendiamo il blocco di indirizzi da controllare e facciamo una passata di reverse DNS su tutto:
- **nmap -sL 172.17.69.0/24**

Setacciare gli IP da analizzare

- nmap non utilizza le librerie di sistema per mandare le richieste DNS, utilizza un suo sistema interno (causa colli di bottiglia).
- Riesce a parallelizzare meglio le richieste.
- Ricordate che le richieste rDNS vanno a finire comunque verso un server dell'organizzazione, quindi potrebbero accorgersi della miriade di richieste (mah..)
- Con --**dns-servers** <server1>,<server2>
  - Possiamo fare override dei DNS di sistema

Host discovery: modalità

- List scan (**-sL**): È quello che abbiamo visto adesso: rDNS e lista ma niente ping/port scan
- Ping scan (**-sP**): Viene eseguito **solo** rDNS e ping scan (lista degli host up e down), no portscanning
- Disable ping (**-PN**): Salta la parte di ping scan e fai portscan su tutto quello che ti ho dato. Non controllare cosa è up o down.

Tecniche di Host Discovery

- Tutti conosciamo la ICMP ECHO REQUEST: ping
- Per fare host discovery con ICMP: **-PE**
- Bisogna essere root per farlo (**ping usa SUID** o capabilites come ben sapete)

sudo nmap -n -PE -sP 10.188.160.1

```
Starting Nmap 5.21 ( http://nmap.org ) at 2013-04-17 23:14
CEST
Nmap scan report for 10.188.160.1
Host is up (0.0090s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
```

sudo nmap -n -PE -sP 10.188.160.1 -oG -

```
# Nmap 5.21 scan initiated Wed Apr 17 23:14:30 2013 as:
nmap -n -PE -sP -oG - 10.188.160.1
Host: 10.188.160.1 () Status: Up
# Nmap done at Wed Apr 17 23:14:30 2013 -- 1 IP address (1
host up) scanned in 0.15 seconds
```

Tecniche di Host Discovery

- Oltre al ping classico si possono sfruttare messaggi ICMP più esotici:
  - ICMP timestamp **(-PP)**
  - ICMP address mask **(-PM)**

Tecniche di Host Discovery

- **TCP SYN PING**
  - Si manda un TCP SYN sulla porta 80 dell'host remoto, se risponde qualcosa (SYN/ACK, RST) significa che è **up**.
  - Si utilizza con **-PS**<portlist>
  - Ad esempio **-PS22** manda un SYN sulla porta 22
  - Si può combinare con il ping!
    -PE -PS22
    - **Segnalo un host come up se risponde a ping oppure al syn sulla porta tcp 22**
  - Attenzione anche qua al discorso privilegi (root)

Tecniche di Host Discovery

- **TCP ACK Ping**
  - Si manda un pacchetto ACK su una porta tcp (ad esempio l'80).
  - Ma l'ACK di quale connessione? Nessuna. Ergo viene generato un RST e risparatoci in faccia
  - Che vantaggio c'è rispetto ad usare il SYN ping? Se davanti all'host c'è un firewall stateless riusciamo a scavalcarlo (anche se ormai è raro trovarne...).
  - L'opzione è **-PA**<port list> e anche in questo caso possiamo specificare tipo **-PA80**
  - Lo svantaggio è che dei firewall stateful potrebbero bloccare!

Tecniche di Host Discovery

- **UDP ping**
  - Si abilita in modo analogo con **-PU**<portlist>
  - Manda un pacchetto UDP vuoto ad una data porta (di default 31338, una porta a caso probabilmente non “aperta”)
  - L'host dovrebbe rimandare indietro un ICMP port unreachable, rivelando il fatto di essere up.

Tecniche di Host Discovery

- **IP protocol ping**
  - In IPv4 un datagramma può contenere diversi protocolli di livello 4: TCP, UDP, ICMP, ma anche Ipsec, IGMP, IP-in-IP.... Per scegliere il protocollo da inserire si utilizza un numero di protocollo (simile al numero di porta), nell'header IP.
  - Per standard, se mandiamo i pacchetti su un protocollo non abilitato, deve tornarci indietro ICMP protocol unreachable
  - Quindi mandiamo richieste su protocolli non abilitati e controlliamo le risposte
  - Si abilita con **-PO**<protocol list>
  - Non funziona sureti nattate :)

Tecniche di Host Discovery

- **ARP Scan**
  - Il tipo di host discovery che funziona quasi nel 99% dei casi IMHO, però può essere usato solo in **reti locali**.
  - Quando siamo in una LAN, per mandare un pacchetto ad un host dobbiamo scoprire il suo MAC address a partire dall'IP, usando una richiesta ARP:
    - Chi ha l'ip 192.168.0.5?
    - IO! sono DE:AD:BE:EF:CA:FE !
  - Quindi mandiamo brutalmente richieste ARP e controlliamo se risponde qualcuno
  - Problema: il processing delle richieste ARP è **lento**!
  - Brutte cose succedono quando si riempie la arp table
  - Quindi nmap fa da sè

```
~$ sudo nmap -PR -sP -n 192.168.0.50

Starting Nmap 5.21 ( http://nmap.org ) at 2013-04-18 00:00
CEST
Nmap scan report for 192.168.0.50
Host is up (0.0020s latency).
MAC Address: 00:19:5B:0F:2B:A6 (D-Link)
Nmap done: 1 IP address (1 host up) scanned in 0.15
seconds
```

Tecniche di Host Discovery

- Quando non scriviamo niente come opzioni in nmap, vengono utilizzate quelle di default:
  - **-PE -PS443 -PA80 -PP** su reti remote
  - **-PR** (ARP) aggiunta su reti locali in automatico

Non c'è scritto nessuna opzione di host discovery = prendo quelle di default.

```
nmap 10.16.78.12
```

C'è l'opzione per ECHO request, uso SOLO quella.

```
nmap -PE 10.16.78.12
```

Port scanning

- Nmap lavora con TCP e UDP per fare port scanning.
- Ricordiamoci: una connessione è identificata dalla quintupla:
  - **SrcIP,SrcPort,DstIP,DstPort,Protocol**
- Le porte sono a 16 bit: vanno da 0 a 65535
- In teoria la porta 0 è invalida.

```
~$ cat /proc/sys/net/ipv4/ip_local_port_range
32768 61000
```

Port scanning: Tipi di porte

- Well-known (1-1023):
  - Porte notissime: 22,80,25,...
- Registered (1024-49151)
  - Anche queste registrate tramite la IANA
- Dynamic e private ports (49152-65535)

Port scanning: Porte **TCP** più comunemente aperte

- (basato su scan del 2008 da parte dell'autore di nmap:
  - 80
  - 23
  - 443
  - 21
  - 22
  - 25
  - 3389
  - ...
  - 110
  - 445
  - 139
  - 143
  - 53
  - 135
  - 3306
  - ...

Port scanning: Porte **UDP** più comunemente aperte

- (basato su scan del 2008)
  - 631 (internet printing protocol)
  - 161 (SNMP)
  - 137 (netbios)
  - 123 (NTP)
  - 138 (netbios)
  - 1434 (microsoft mysql)
  - ..
  - **1900 (UpnP)**
  - **...**

Port scanning: stati delle porte

- **`open`**
- **`closed`**
- **`filtered`**
- **`unfiltered`**
- **`open|filtered`**
- **`closed|filtered`**

Port scanning

- Di default viene fatto scanning sulle 1000 porte più comuni. Ovviamente lo scanning non viene fatto in modo sequenziale ma random.
- Di default si fa solo scan TCP

```
~# nmap scanme.nmap.org

Interesting ports on scanme.nmap.org (74.207.244.221):
Not shown: 994 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
25/tcp  closed smtp
53/tcp  open   domain
70/tcp  closed gopher
80/tcp  open   http
113/tcp closed auth

Nmap done: 1 IP address (1 host up) scanned in 2.23
seconds
```

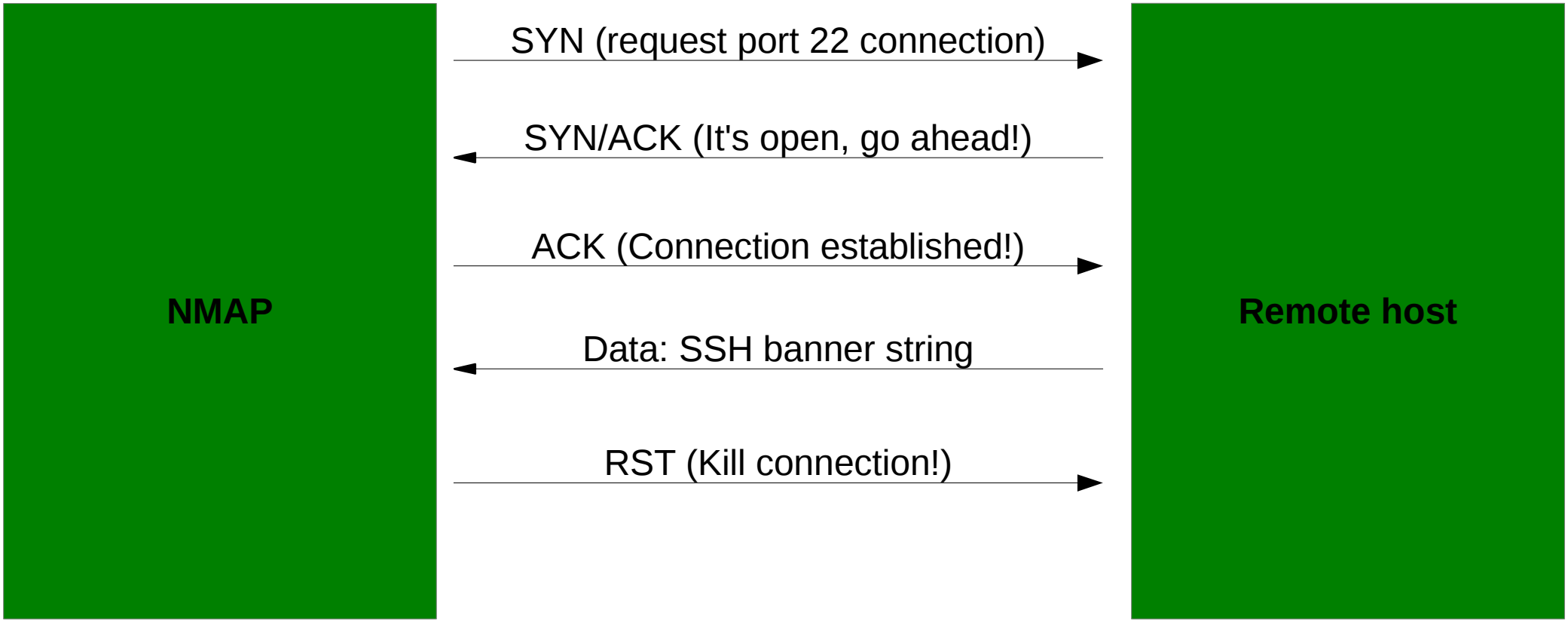
Port scanning

- Di default si fa solo scan TCP utilizzando il tipo di scan **“TCP Connect Scan”** (**-sT**). Questo tipo di scan fa uso della syscall **connect()** da utente non privilegiato
- Però è molto meglio usare **-sS** per fare il **SYN Scan**. Con il syn scan mandiamo direttamente pacchetti syn a basso livello, senza aprire una vera connessione.
- Cosa cambia?

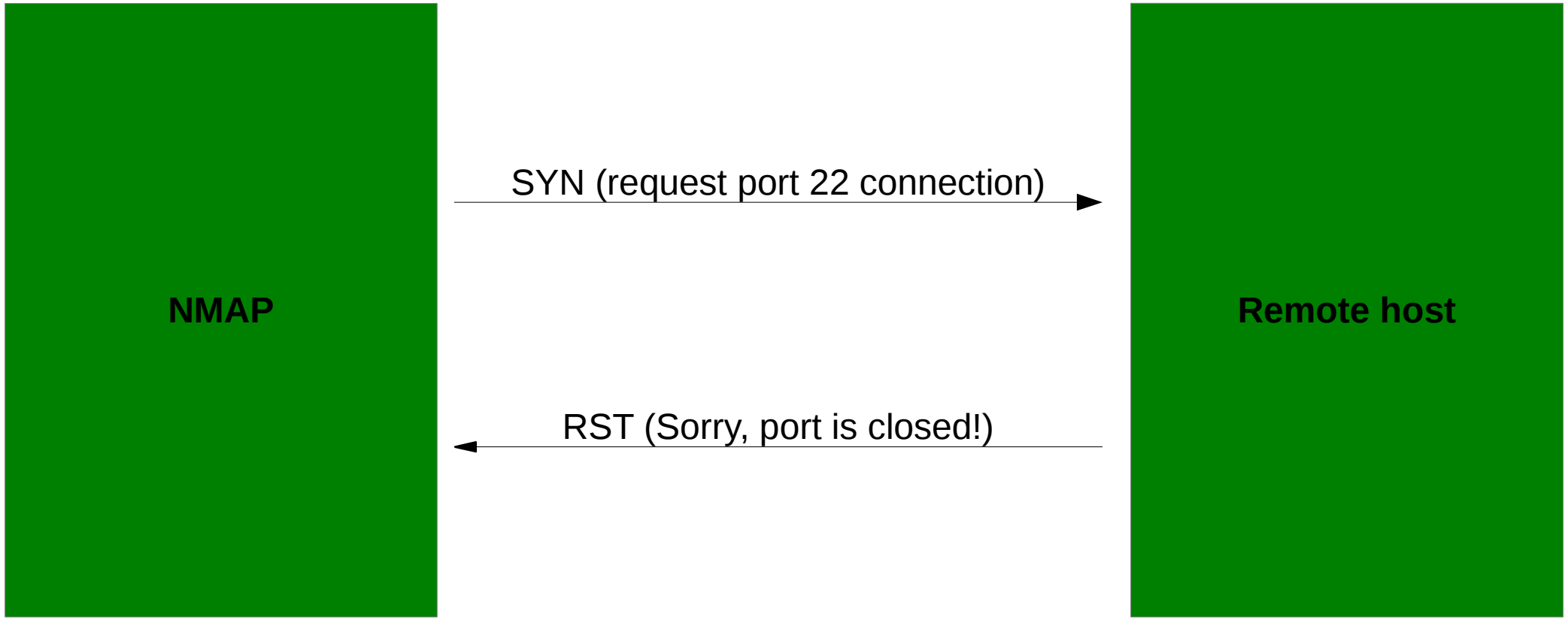
Differenza ConnectScan e SynScan

- Meglio il Syn Scan, però richiede di essere **root**.
- Ora vediamo perché.

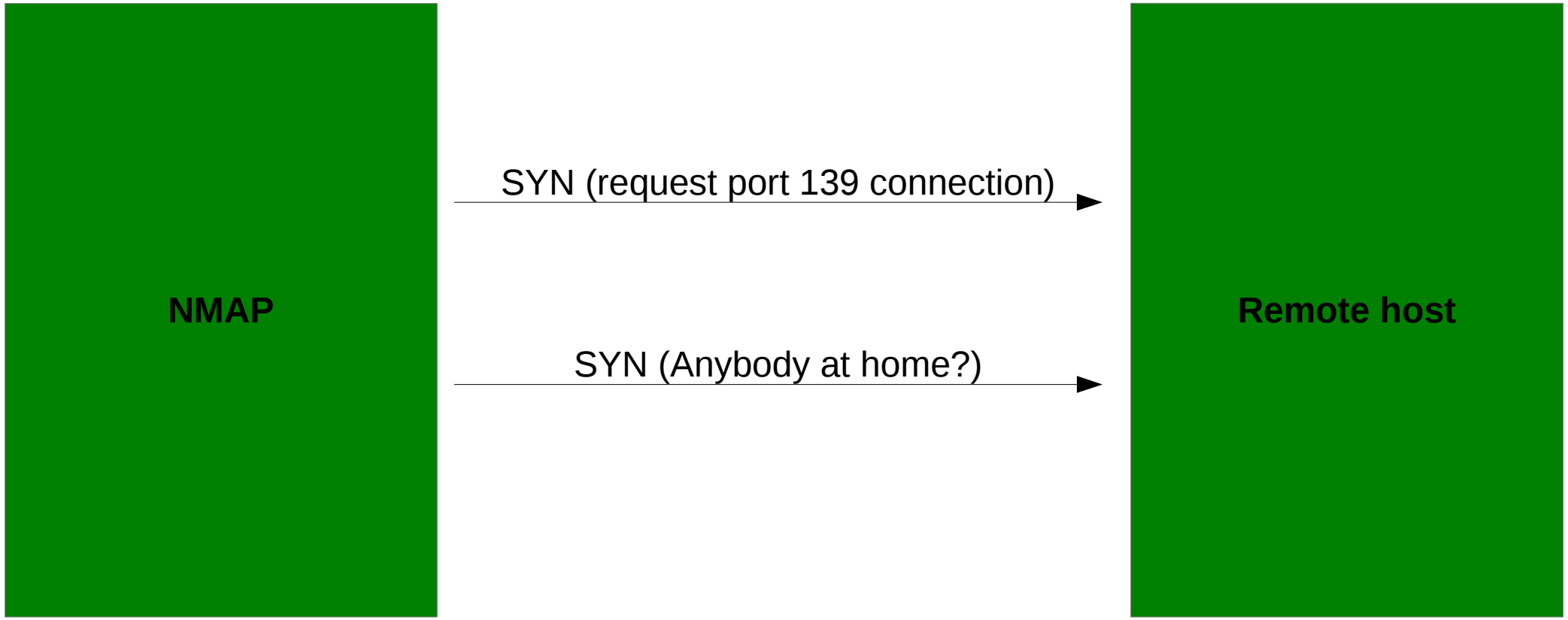
ConnectScan (-sT)



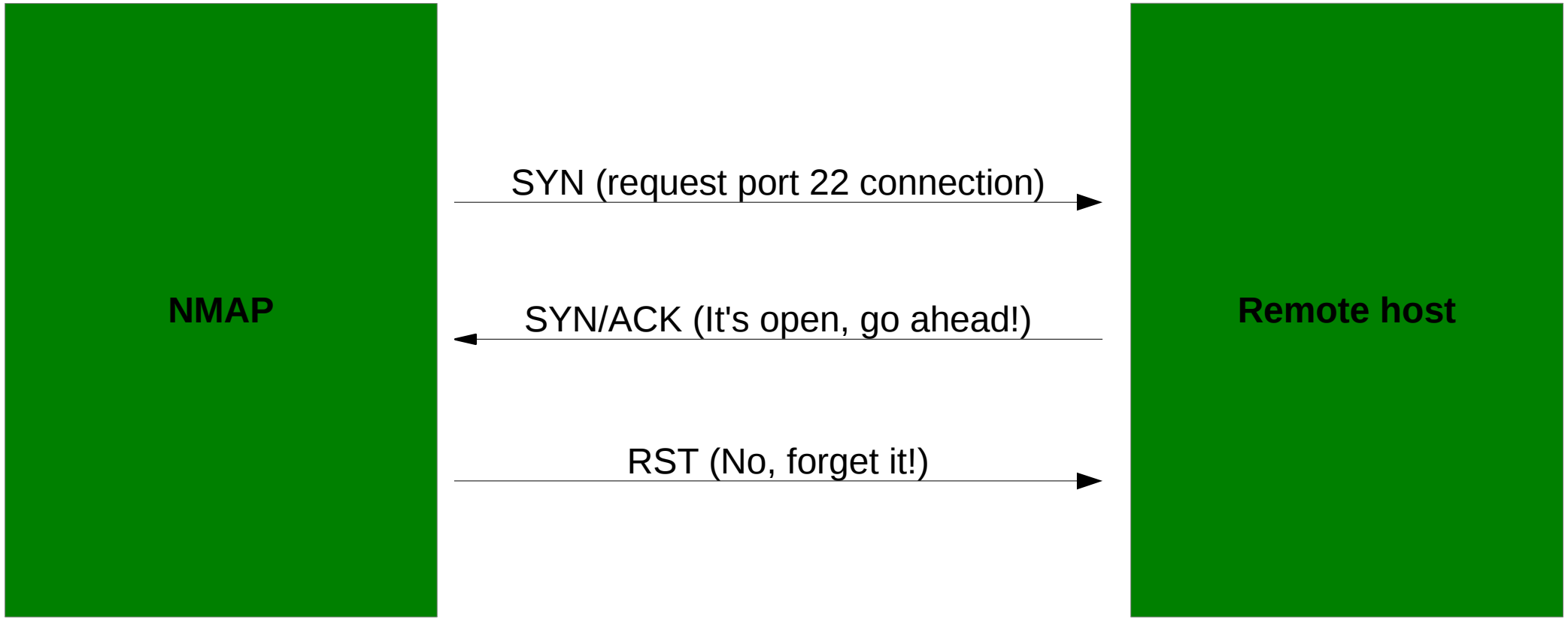
ConnectScan (-sT)



ConnectScan (-sT)



SYN scan (-sS)


NMAP
SYN (request port 22 connection)
SYN/ACK (It's open, go ahead!)
RST (No, forget it!)
Remote host

Differenza ConnectScan e SynScan

- Meglio il Syn Scan, però richiede di essere **root**.
- Nmap ha meno controllo usando connect scan, è meno efficiente.
- Il connect scan non solo impiega più tempo, ma rischia anche di far loggare la connessione.
- In entrambi i casi le risposte vengono interpretate così:
  - Risposta TCP SYN/ACK → Open
  - Risposta TCP RST → Closed
  - Nessuna Risposta → Filtered
  - ICMP unreachable → Filtered

Specificare le porte

- **-p**<portlist>
- Ad esempio **-p22,110,25** oppure **-p0-65535**
- Se al posto di **-p** usiamo **-F**, vengono portscannate le prime 100 porte più comuni.
-

UDP Scan

- In UDP non abbiamo i meccanismi SYN,ACK,... etc di TCP, quindi è un po' più compilcato capire se una porta è aperta.
- Il fatto che sia più complicato non significa che bisogna dimenticarle!
- Negli UDP portscans viene mandato un pacchetto UDP **vuoto** sulle porte scelte

UDP Scan

- Quindi mandiamo questo pacchetto UDP vuoto, le risposte sono:
  - Una qualunque risposta UDP (insolito) → **open**
  - Nessuna risposta ricevuta → **open|filtered**
  - ICMP port unreachable → **closed**
  - Altri errori ICMP → **filtered**

```
~# nmap -sU -p0-1024 otacon22.it

Starting Nmap 5.00 ( http://nmap.org ) at 2013-04-18
09:42 CEST
All 1025 scanned ports on ks27755.kimsufi.com
(91.121.88.107) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 37.79
seconds
```

UDP Scan

- Non essendoci quasi mai nessuna risposta, è un tipo di scan quasi sempre lento
- C'è un grave limite sulla frequenza di pacchetti ICMP che escono dall'host destinazione
  - cat /proc/sys/net/ipv4/icmp_ratelimit
  - 1000 (← millisecondi tra due pacchetti ICMP di risposta)
- Uno scan di 65535 porte può **impiegare 18 ore**.
  - Possiamo risolvere in parte parallelizzando su più host e facendo scan prima sulle porte più comuni, usando --host-timeout
- Se non vi serve, non abilitatelo, perdete solo tanto tempo.
- Come risolvere il problema delle porte
  - open|filtered ?
  - Lo vediamo adesso con il service scan.

Ancora scan TCP: FIN, NULL,Xmas

- La storica RFC793 di TCP afferma:
  - *“If the [destination] port state is CLOSED …. an incoming segment not containing a RST causes a RST to be sent in response”*
- Sfruttando questa proprietà si tenta di generare delle risposte
- Null scan: **-sN**
- FIN scan: **-sF**
- Xmas scan: **-sX**

Ancora scan TCP: FIN, NULL,Xmas

- Comportamenti a seconda delle risposte:
  - Nessuna risposta ricevuta → **open|filtered**
  - TCP RST packet → **closed**
  - ICMP unreachable → **filtered**
- Il vantaggio di questo tipo di scan è che riescono a scavalcare alcuni firewall non-stateful che filtrano i SYN o ACK.
- Il problema è che alcuni OS (windows, cisco...) non rispettano lo standard e mandano RST anche per porte aperte (vediamo poi che si può sfruttare l'OS fingerprinting per capirlo).

Ancora scan TCP: FIN, NULL,Xmas

- Potete forgiare il vostro tipo di scan usando --scanflags
  - **--scanflags SYNFIN**
- Lo scan SYN/FIN ha un certo senso

Ancora scan TCP: ACK scan

- L'ACK scan **(-sA)** è un po' diverso dai precedenti per come interpreta le risposte:
  - TCP RST Response → **unfiltered**
  - Nessuna risposta → **filtered**
  - ICMP unreachable → **filtered**
-

Ancora scan TCP: TCP window scan

- Il Window scan **(-sW)** sfrutta un dettaglio implementativo di alcuni stack TCP/IP dove:
  - Funziona come ACK scan: manda degli ACK
  - Quando arrivano risposte analizza la dimensione della finestra. In alcuni OS la finestra del TCP è positiva per porte aperte e zero per porte chiuse
- Quindi:
  - TCP RST response con finestra non-zero → **open**
  - TCP RST response con finestra zero → **closed**
  - Nessuna risposta → **filtered**
  - ICMP unreachable error → **filtered**

Ancora scan TCP: TCP window scan

- Non è un tipo di scan “affidabile”
- Su alcuni host il comportamento è esattamente l'opposto.
- Anche qua si consiglia di fare prima OS detection.
-

Ancora scan TCP: Seghe mentali – IDLE scan

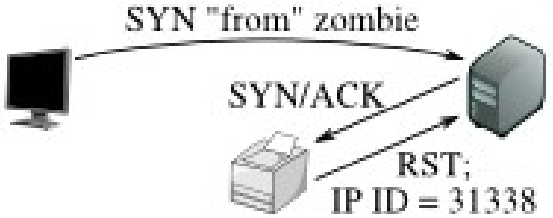
- L'IDLE Scan **(-sI)** sfrutta un dettaglio implementativo di molti stack TCP/IP, riguardo all'identificatore del campo IP.
- È molto lento e non funziona sempre, ma è **molto furtivo**.
- Nel datagramma IP c'è un identification number. In molti stack viene semplicemente implementato per ogni pacchetto che viene inviato. È un contatore globale a livello di sistema operativo

Ancora scan TCP: Seghe mentali – IDLE scan

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.

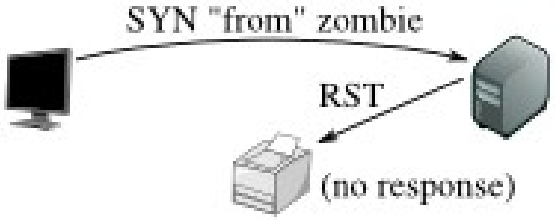
The zombie's IP ID has increased by 2 since step 1, so the port is open!

Ancora scan TCP: Seghe mentali – IDLE scan

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID. This step is always the same.

Step 2: Forge a SYN packet from the zombie.



The target sends a RST (the port is closed) in response to the SYN that appears to come from the zombie. The zombie ignores the unsolicited RST, leaving its IP ID unchanged.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by only 1 since step 1, so the port is not open.

Ancora scan TCP: Seghe mentali – IDLE scan

Step 1: Probe the zombie's IP ID.

Just as in the other two cases, the attacker sends a SYN/ACK to the zombie. The zombie discloses its IP ID.

Step 2: Forge a SYN packet from the zombie.

The target, obstinately filtering its port, ignores the SYN that appears to come from the zombie. The zombie, unaware that anything has happened, does not increment its IP ID.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by only 1 since step 1, so the port is not open. From the attacker's point of view this filtered port is indistinguishable from a closed port.

Ancora scan TCP: Seghe mentali – IDLE scan

- Usando la OS detection ci vengono date informazioni sul modo di incrementare l'IP identification number e quindi troviamo gli zombie per l'attacco.
- È molto probabile che ci siano errori se l'host comunica mentre stiamo scannando
- C'è un sistema di ricerca binaria per trovare la porta aperta facendo una passata su tante porte insieme (per risparmiare tempo)

IP Protocol Scan (**-sO**)

- L'abbiamo già visto nella parte di host detection. Viene usato per scoprire quali protocolli sono supportati dall'host remoto
- Si mandano dei pacchetti vuoti sulle varie porte di protocollo e il comportamento è:
  - Qualunque risposta dall'host → **open**
  - ICMP protocol unreachable error → **closed**
  - Altri errori ICMP → **filtered**
  - Nessuna risposta → **open|filtered**
-

TCP FTP Bounce Scan (**-b**)

- Si sfrutta una funzionalità di FTP active mode
- In FTP active mode con il comando **PORT** dite al server: mandami questo file su questo IP e questa porta.
- Se il server non riesce a collegarsi vi da un errore sulla vostra connessione FTP.
- Quindi il server FTP funziona da “proxy”
- È un buon modo per nascondere la sorgente di uno scan, anche se abbastanza lento
- Non tutti i server FTP lo supportano :) (molti controllano l'IP destinazione == IP connessione FTP).

Network Condition Monitoring

- Nmap non **spara** pacchetti al massimo della velocità (non è una buona idea)
- Contiene al suo interno un sistema di controllo della congestione simile a quello standard di TCP.

Network Condition Monitoring

- Dimensione dell'hostgroup:
  - **--min-hostgroup , --max-hostgroup**
- Numero di probes lanciate in parallelo:
  - **--min-parallelism , --max-parallelism**
- Numero massimo di ritrasmissioni di una probe:
  - **--max-retries**
- Massimo tempo di timeout:
  - **--host-timeout**
- Ritardo inserito tra ogni probe verso un singolo host:
  - **--min-rate , --max-rate**

Network Condition Monitoring

- Esistono una serie di **Timing Templates** precotti dagli sviluppatori:
  - **-T0** Paranoid
  - **-T1** Sneaky
  - **-T2** Polite
  - **-T3** Normal (default)
  - **-T4** Aggressive
  - **-T5** Insane

Lentissimi ma possono essere utili contro IDS

Consigliato se siete sulla stessa LAN o a pochi hop di distanza

Network Condition Monitoring: differenze tra timings

	T0	T1	T2	T3	T4	T5
--initial-rtt-timeout	300'000	15'000	1'000	1'000	500	250
--max-retries	10	10	10	10	6	2
--scan-delay	300'000	15'000	400	0	0	0
--min-parallelism	Dimanico, non dipende dai templates					
--max-parallelism	1	1	1	Dinamico	Dinamico	Dinamico
--min-hostgroup	Dimanico, non dipende dai templates					
--max-hostgroup	Dimanico, non dipende dai templates					
--min-rate	No limit minimo					
--max-rate	No limit massimo					

Service scanning

- Si abiliata con **-sV**
- Viene lanciato sulle porte che vengono trovate aperte durante la fase di port scanning.
-

```
# nmap -A -T4 localhost

Starting Nmap ( http://nmap.org )
Nmap scan report for felix (127.0.0.1)
(The 1640 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE    VERSION
21/tcp   open  ftp        WU-FTPD wu-2.6.1-20
22/tcp   open  ssh        OpenSSH 3.1p1 (protocol 1.99)
53/tcp   open  domain     ISC BIND 9.2.1
79/tcp   open  finger     Linux fingerd
111/tcp  open  rpcbind    2 (rpc #100000)
443/tcp  open  ssl/http   Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-
dev)
515/tcp  open  printer
631/tcp  open  ipp        CUPS 1.1
953/tcp  open  rndc?
5000/tcp open  ssl/ftp    WU-FTPD wu-2.6.1-20
5001/tcp open  ssl/ssh    OpenSSH 3.1p1 (protocol 1.99)
5002/tcp open  ssl/domain ISC BIND 9.2.1
5003/tcp open  ssl/finger Linux fingerd
6000/tcp open  X11        (access denied)
8000/tcp open  http-proxy Junkbuster webproxy
8080/tcp open  http       Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-
dev)
8081/tcp open  http       Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-
dev)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20

Nmap finished: 1 IP address (1 host up) scanned in 42.494 seconds
```

#Esempio di probe

```
# The Exclude directive takes a comma separated list of ports.
# The format is exactly the same as the -p switch.
Exclude T:9100-9107

# This is the NULL probe that just compares any banners given to us
##############################NEXT PROBE##############################
Probe TCP NULL q||
# Wait for at least 5 seconds for data.  Otherwise an Nmap default is
used.
totalwaitms 5000
# Windows 2003
match ftp m/^220[ -]Microsoft FTP Service\r\n/ p/Microsoft ftpd/
match ftp m/^220 ProFTPD (\d\S+) Server/ p/ProFTPD/ v/$1/
softmatch ftp m/^220 [-.\w ]+ftp.*\r\n$/i
match ident m|^flock\(\) on closed filehandle .*midentd| p/midentd/
i/broken/
match imap m|^\* OK Welcome to Binc IMAP v(\d[-.\w]+)| p/Binc IMAPd/
v$1/
softmatch imap m/^\* OK [-.\w ]+imap[-.\w ]+\r\n$/i
match lucent-fwadm m|^0001;2$| p/Lucent Secure Management Server/
match meetingmaker m/^\xc1,$/ p/Meeting Maker calendaring/
# lopster 1.2.0.1 on Linux 1.1
match napster m|^1$| p/Lopster Napster P2P client/

Probe UDP Help q|help\r\n\r\n|
rarity 3
ports 7,13,37
match chargen m|@ABCDEFGHIJKLMNOPQRSTUVWXYZ|
match echo m|^help\r\n\r\n$|
```

Service scanning

- Quindi la stessa probe può essere applicata su tante porte diverse
- Potete utilizzare un vostro database di probe o modificare l'esistente
- Potete submittare probes per nuovi servizi e saranno incluse nella prossima release di nmap ;)
- Ricordiamoci quanto sono utili per probes per UDP, oltre per scoprire che un servizio TCP è su una porta non standard.
-

OS fingerprinting

- Nmap integra una funzionalità di OS fingerprinting.
- Vengono inviati una serie di pacchetti strutturati in modo particolare e si osservano diversi bit specifici nelle risposte. A seconda del comportamento si matcha su un database e si scopre il sistema operativo
- Spesso il livello di matching è preciso: si arriva fino alla versione del kernel linux o Windows, Mac!
- Può essere usato per sgamare AP rogue e macchine non aggiornate! Oppure per social engineering
- La tecnica viene spesso usata insieme al service discovery perché aiuta anche a capire se c'è un firewall o altro in mezzo

```
# nmap -O -v scanme.nmap.org

Starting Nmap ( http://nmap.org )
Nmap scan report for scanme.nmap.org (74.207.244.221)
Not shown: 994 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    open     http
646/tcp   filtered ldp
1720/tcp  filtered H.323/Q.931
9929/tcp  open     nping-echo
31337/tcp open     Elite
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.39
OS details: Linux 2.6.39
Uptime guess: 1.674 days (since Fri Sep  9 12:03:04 2011)
Network Distance: 10 hops
TCP Sequence Prediction: Difficulty=205 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/local/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.58 seconds
           Raw packets sent: 1063 (47.432KB) | Rcvd: 1031 (41.664KB)
```

OS fingerprinting

- Device type
- Running
- OS details (meglio avere porta TCP aperta e chiusa)
- Uptime
- Network Distance
- TCP Sequence prediction (per fare spoofing utile)
- IP ID sequence generation

Tecniche per OS fingerprinting

- Pacchetti IP per controllare sequence generation
- ICMP echo
- TCP ECN
- 7 pacchetti TCP diversi
- Roba UDP
- ...

OS fingerprinting

- Viene utilizzato un database per confrontare le varie risposte
- Viene fornito un match probabilistico se non matcha completamente.
- Se non c'è match c'è possibilità di submittare il fingerprint alla community

scripts

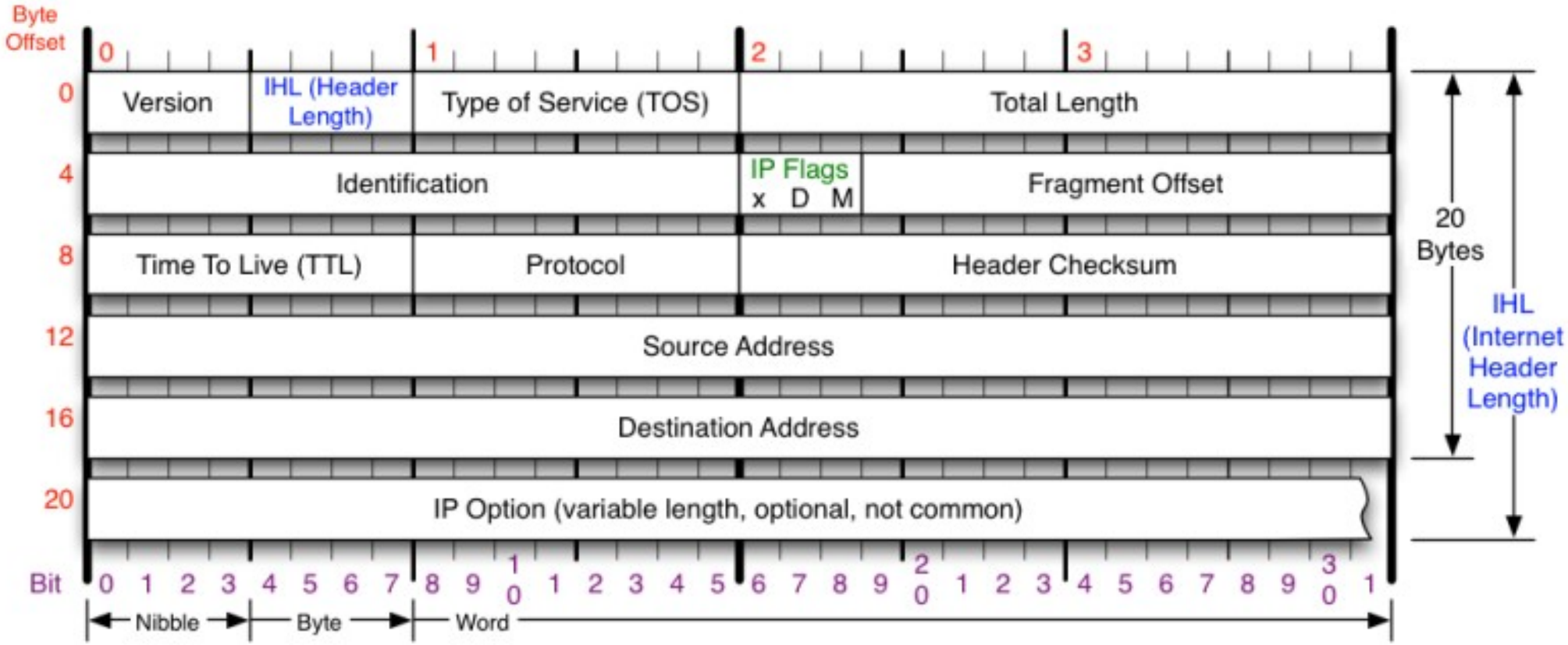
- **nmap –script=./showSSHVersion.nse 127.0.0.1**
- Esempi di scripts:
  - Controllo su DNS recursion, DNS zone transfer,
  - Controllo FTP bounce
  - Tirare fuori IRC info di un server
  - Tirare fuori info su un server FTP, Mysql
  - Match contro la versione 2 di skype.
  - Lista delle cartelle condivise su netbios/samba, oppure lista utenti
- Si tratta comunque sempre di **scanning**! Per gli attacchi c'è metasploit ;)

Sorpassare IDS e firewalls

- Trucchettazzo IP ID con i firewalls per controllare gli IP bloccati.
- RALLENTARE è la regola d'oro con gli IDS...
- Scherzacci forzando la frammentazione IP

Domande?

- Contact me: **rettore at polimi.eu**



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

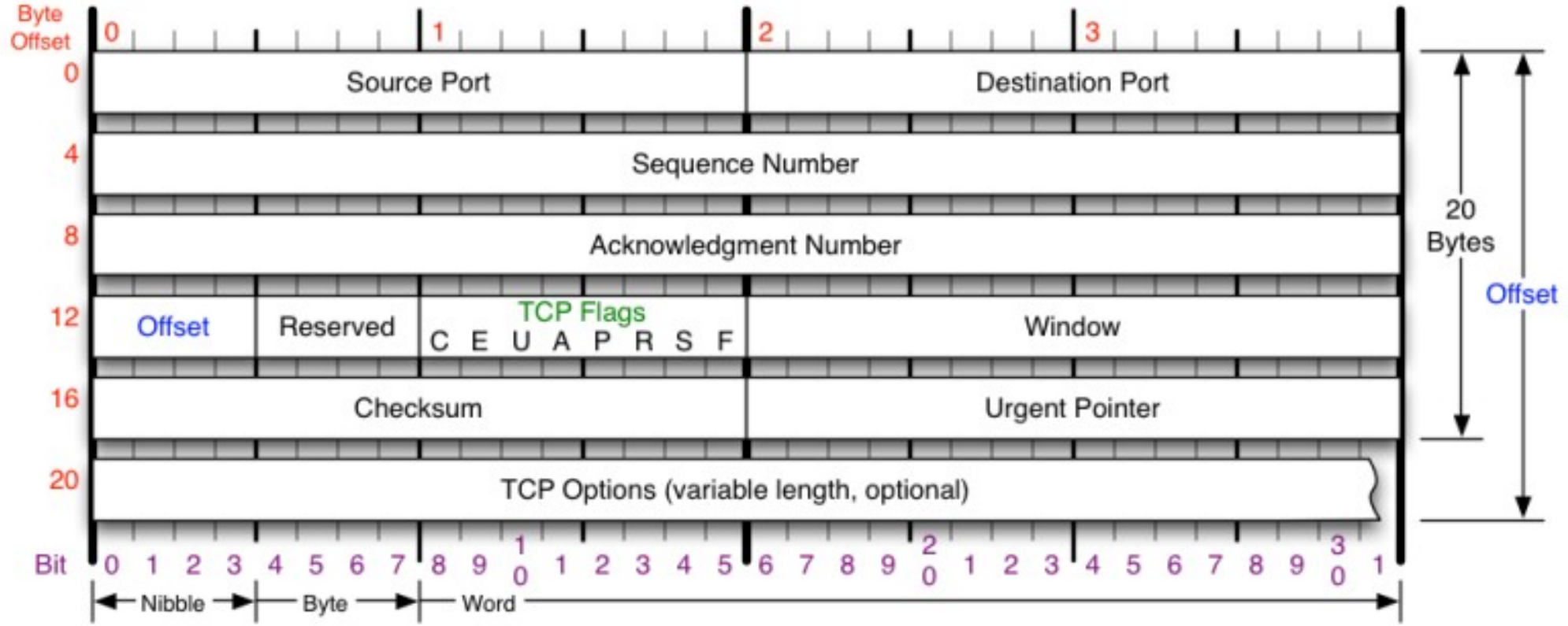
IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.