Università degli Studi di Milano
Facoltà di Scienze Matematiche, Fisiche e Naturali
Anno Accademico 2013/2014

Attacchi di rete

Andrea Lanzi

4 Giugno 2013

Background

IPv4 address

Ogni *host* possiede un proprio indirizzo IP (32 bit) generalmente nella forma: `IP` = `NetId` - `SubnetId` - `HostId`

Subnet mask

La *subnet mask* definisce il confine (perimetro) tra *host* e *NetId* - *SubnetId*.

CIDR (Classless Inter-Domain Routing)

`IP` = $w.x.y.z/n$

n: indica il numero di bit che compongono *NetId* - *SubnetId*

Esempio

NetId - *SubnetId* = 192.168.1.0

Subnet mask = 255.255.255.0

Notazione *CIDR* = 192.168.1.0/24

Background

IPv4 address

Ogni *host* possiede un proprio indirizzo IP (32 bit) generalmente nella forma: `IP` = `NetId` - `SubnetId` - `HostId`

Subnet mask

La *subnet mask* definisce il confine (perimetro) tra *host* e *NetId* - *SubnetId*.

CIDR (Classless Inter-Domain Routing)

`IP` = $w.x.y.z/n$

n: indica il numero di bit che compongono *NetId* - *SubnetId*

Esempio

NetId - *SubnetId* = 192.168.1.0

Subnet mask = 255.255.255.0

Notazione *CIDR* = 192.168.1.0/24

Background

IPv4 address

Ogni *host* possiede un proprio indirizzo IP (32 bit) generalmente nella forma: `IP` = `NetId` - `SubnetId` - `HostId`

Subnet mask

La *subnet mask* definisce il confine (perimetro) tra *host* e *NetId* - *SubnetId*.

CIDR (Classless Inter-Domain Routing)

`IP` = $w.x.y.z/n$

n: indica il numero di bit che compongono *NetId* - *SubnetId*

Esempio

NetId - *SubnetId* = 192.168.1.0

Subnet mask = 255.255.255.0

Notazione *CIDR* = 192.168.1.0/24

Background

IPv4 address

Ogni *host* possiede un proprio indirizzo IP (32 bit) generalmente nella forma: `IP` = `NetId` - `SubnetId` - `HostId`

Subnet mask

La *subnet mask* definisce il confine (perimetro) tra *host* e *NetId* - *SubnetId*.

CIDR (Classless Inter-Domain Routing)

`IP` = $w.x.y.z/n$

n: indica il numero di bit che compongono *NetId* - *SubnetId*

Esempio

NetId - *SubnetId* = 192.168.1.0

Subnet mask = 255.255.255.0

Notazione *CIDR* = 192.168.1.0/24

Anatomia di un attacco

1. ☐ determinare macchine attive
2. ☐ determinare sistema operativo utilizzato
3. ☐ determinare servizi in esecuzione
4. ☐ *exploit* dei servizi vulnerabili

Anatomia di un attacco

1. ☒ determinare macchine attive
2. ☒ determinare sistema operativo utilizzato
3. ☒ determinare servizi in esecuzione
4. ☐ *exploit* dei servizi vulnerabili

Anatomia di un attacco

1. ☒ determinare macchine attive
2. ☒ determinare sistema operativo utilizzato
3. ☒ determinare servizi in esecuzione
4. ☐ *exploit* dei servizi vulnerabili

Osservazioni

Non tutti gli attacchi seguono i passi precedenti...

Host discovery

Obiettivo

- identificare host attivi sulla (sotto)rete
- focalizzare l'attenzione sugli host "interessanti" (e.g., solo i laptop, solo apparati di reti, ...)
- minima quantità di traffico

Come?

1. PING scan (ICMP/ARP/UDP/...)
2. reverse-DNS (utili informazioni spesso recuperate dai nomi degli host)
3. DB indirizzi MAC

Host discovery

Utility di sistema:

- whois
- host
- ping

DNS:

- nslookup
- dig

NMAP host specification

Individuati gli indirizzi IP o host è necessario indicare su quali macchine si desidera fare lo scan.

3 possibilità:

- `-iL`: "Input From List"
- `-iR <numtargets>`: "Choose Targets at Random"
- `--exclude, --excludefile <filename>`: "Excluding Targets"

Esempi:

nmap scanme.nmap.org , nmap scanme.nmap.org/32 , nmap 64.13.134.52
nmap 64.13.134.52/24 −−exclude scanme.nmap.org,insecure.org
nmap 10.0.0.0/8 −−exclude 10.6.0.0/16,ultra-sensitive-host.company.com
egrep '^lease' /var/lib/dhcp/dhcpd.leases | awk 'print $2' | nmap -iL -

List Scan (-sL)

Lista tutti gli host sulla rete/i specificata/e *senza inviare* nessun pacchetto agli host.
Reverse-DNS viene effettuata per trovare i nomi degli host associati a ciascun indirizzo IP

Esempio:

```
[12:08:41]srdjan@salvador~:$ nmap -sL 192.168.1.0/29

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 12:08 CEST
Nmap scan report for 192.168.1.0
Nmap scan report for 192.168.1.1
Nmap scan report for 192.168.1.2
Nmap scan report for 192.168.1.3
Nmap scan report for 192.168.1.4
Nmap scan report for 192.168.1.5
Nmap scan report for 192.168.1.6
Nmap scan report for 192.168.1.7
Nmap done: 8 IP addresses (0 hosts up) scanned in 4.14 seconds
```

PING scan (-sn)

- **privileged user**:
  - ICMP `echo req.`, TCP SYN, TCP ACK, ICMP `timestamp req.`
  - ARP requests (su local ethernet network)
- **un-privileged user**:
  - TCP SYN verso porta 80 e porta 443

Esempio

```
root@salvador:/home/srdjan# nmap -sn 192.168.1.2

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-11 14:28 CEST
Nmap scan report for kebab.laser.di.unimi.it (159.149.148.7)
Host is up (0.00020s latency).
MAC Address: 08:00:27:A3:EE:36 (3Com)
Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

Traffico generato

```
root@sec1:~# tcpdump -ni eth1 port not 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
15:31:57.848664 ARP, Request who-has 192.168.1.2 (ff:ff:ff:ff:ff:ff) tell 192.168.1.3, length 46
15:31:57.848726 ARP, Reply 192.168.1.2 is-at 08:00:27:a3:ee:36, length 28
```

PING scan (-sn)

- **privileged user**:
  - ICMP `echo req.`, TCP SYN, TCP ACK, ICMP `timestamp req.`
  - ARP requests (su local ethernet network)
- **un-privileged user**:
  - TCP SYN verso porta 80 e porta 443

Esempio

```
root@salvador:/home/srdjan# nmap -sn 192.168.1.2

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-11 14:28 CEST
Nmap scan report for kebab.laser.di.unimi.it (159.149.148.7)
Host is up (0.00020s latency).
MAC Address: 08:00:27:A3:EE:36 (3Com)
Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

Traffico generato

```
root@sec1:~# tcpdump -ni eth1 port not 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
15:31:57.848664 ARP, Request who-has 192.168.1.2 (ff:ff:ff:ff:ff:ff) tell 192.168.1.3, length 46
15:31:57.848726 ARP, Reply 192.168.1.2 is-at 08:00:27:a3:ee:36, length 28
```

Disable Ping (-Pn)

Di default `nmap` effettua uno "ping scan" prima di tipologie di scan più aggressivo come scanning delle porte aperte, identificazione del OS, identificazione delle versioni di software attivo sulla macchina.

Disabilitando la fase di "host discovery" verranno ispezionati tutti gli IP passati in input al programma.

In quali circostanze potrebbe tornare utile disabilitare i "ping scan" di `nmap`?

Disable Ping (-Pn)

Di default `nmap` effettua uno "ping scan" prima di tipologie di scan più aggressivo come scanning delle porte aperte, identificazione del OS, identificazione delle versioni di software attivo sulla macchina.

Disabilitando la fase di "host discovery" verranno ispezionati tutti gli IP passati in input al programma.

In quali circostanze potrebbe tornare utile disabilitare i "ping scan" di `nmap`?

- firewall e pentration testing
- colui che effettua lo scan sa già quali sono le macchine attive/interessanti

TCP SYN Ping (-PS<port list>)

Invia un pacchetto TCP vuoto con il flag SYN impostato (di default alla porta 80).
L'obiettivo è simulare la creazione di una connessione.

- se la porte di destinazione è chiusa, il target risponde RST
- se la porta è aperta target risponde con pacchetto TCP SYN/ACK. Kernel della macchina su cui è usato nmap risponde con RST e chiude la connessione.

Esempio

```
[16:43:33]srdjan@salvador~:$ nmap -sP -PS2345 -v 192.168.1.0/30
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 16:44 CEST
Initiating Ping Scan at 16:44
Scanning 4 hosts [1 port/host]
Completed Ping Scan at 16:44, 0.01s elapsed (4 total hosts)
Initiating Parallel DNS resolution of 4 hosts. at 16:44
Completed Parallel DNS resolution of 4 hosts. at 16:44, 0.17s elapsed
Nmap scan report for 192.168.1.0 [host down]
Nmap scan report for 192.168.1.1
Host is up (0.0044s latency).
Nmap scan report for 192.168.1.2
Host is up (0.011s latency).
Nmap scan report for 192.168.1.3
Host is up (0.000082s latency).
Read data files from: /usr/bin/../share/nmap
Nmap done: 4 IP addresses (3 hosts up) scanned in 0.18 seconds
```

TCP ACK Ping (-PA<port list>)

Simile a TCP SYN Ping; l'unica differenza è che viene impostato il flag ACK invece di SYN.
Si finge di inviare un ACK per una connesione *inesistente*, di conseguenza la macchina target, qualora attiva, dovrebbe rispondere con un pacchetto RST.

SYN o ACK?

- **stateless firewall**: blocco di pacchetti SYN
- **stateful firewall**: blocco di pacchetti "INVALID"

Perciò quale opzione conviene usare?
ENTRAMBE! :)

TCP ACK Ping (-PA<port list>)

Simile a TCP SYN Ping; l'unica differenza è che viene impostato il flag ACK invece di SYN.
Si finge di inviare un ACK per una connesione *inesistente*, di conseguenza la macchina target, qualora attiva, dovrebbe rispondere con un pacchetto RST.

SYN o ACK?

- **stateless firewall**: blocco di pacchetti SYN
- **stateful firewall**: blocco di pacchetti "INVALID"

Perciò quale opzione conviene usare?
ENTRAMBE! :)

TCP ACK Ping (-PA<port list>)

Simile a TCP SYN Ping; l'unica differenza è che viene impostato il flag ACK invece di SYN.
Si finge di inviare un ACK per una connesione *inesistente*, di conseguenza la macchina target, qualora attiva, dovrebbe rispondere con un pacchetto RST.

SYN o ACK?

- **stateless firewall**: blocco di pacchetti SYN
- **stateful firewall**: blocco di pacchetti "INVALID"

Perciò quale opzione conviene usare?
ENTRAMBE! :)

UDP Ping (-PU<port list>)

Invia pacchetti vuoti di tipo UDP alle porte indicate (di default porta 40125).

Quale *ben noto* servizio è in ascolto su queste porta?

UDP Ping (-PU<port list>)

Invia pacchetti vuoti di tipo UDP alle porte indicate (di default porta 40125).

Quale *ben noto* servizio è in ascolto su queste porta?
NESSUNO! Tipicamente UDP non riceve pacchetti in risposta; l'obiettivo è farsi restituire un pacchetto ICMP `port unreachable`.

UDP Ping (-PU<port list>)

ESEMPIO:

IP: 192.168.1.3 **host:** salvador (debian SID)
IP: 192.168.1.8 **host:** openpita (openbsd 5.1)

salvador:

```
nmap -sn -PU 192.168.1.8

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 19:08 CEST
Nmap scan report for 192.168.1.8
Host is up (0.10s latency).
MAC Address: 00:14:7C:34:2F:D3 (3Com)
Nmap done: 1 IP address (1 host up) scanned in 0.36 seconds
```

openpita:

```
# tcpdump -ni em0 host 192.168.1.3
tcpdump: listening on em0, link-type EN10MB
21:08:39.154412 arp who-has 192.168.1.8 (ff:ff:ff:ff:ff:ff) tell 192.168.1.3
21:08:39.154536 arp reply 192.168.1.8 is-at 08:00:27:21:a7:76
21:08:45.401326 192.168.1.3.17500 > 255.255.255.255.17500: udp 212 (DF)
21:08:45.401734 192.168.1.3.17500 > 192.168.1.255.17500: udp 212 (DF)
```

UDP Ping (-PU<port list>)

ESEMPIO:

IP: 192.168.1.3 **host:** salvador (debian SID)
IP: 192.168.1.8 **host:** openpita (openbsd 5.1)

openpita:

```
# nmap -sn -PU 192.168.1.3

Starting Nmap 5.51 ( http://nmap.org ) at 2013-05-09 21:19 CEST
Nmap scan report for 192.168.1.3
Host is up (0.057s latency).
Nmap done: 1 IP address (1 host up) scanned in 5.19 seconds
```

salvador:

```
[19:07:58]root@salvador/home/srdjan:# tcpdump -i wlan0 host 192.168.1.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 65535 bytes
19:08:02.545823 IP 192.168.1.8.57241 > 192.168.1.3.40125: UDP, length 0
19:08:02.545902 IP 192.168.1.3 > 192.168.1.8: ICMP 192.168.1.3 udp port 40125 unreachable, length 36
19:08:03.551902 IP 192.168.1.8.57242 > 192.168.1.3.40125: UDP, length 0
19:08:03.551973 IP 192.168.1.3 > 192.168.1.8: ICMP 192.168.1.3 udp port 40125 unreachable, length 36
```

ICMP ed IP Ping

Oltre a ICMP `echo request` `nmap` è in grado di inviare altri messaggi di tipo ICMP:

- **-PE**: `echo request`
- **-PP**: `timestamp query`
- **-PM**: `address mask query`

È inoltre possibile utilizzare "IP protocol ping" per specificare un particolare protocollo nel header IP.
Obiettivo: ricevere risposte che utilizzino gli stessi protocolli delle richieste, oppure messaggi ICMP che comunicano che il target non supporta il particolare protocollo selezionato.

ARP Scan (-PR)

Usato per scan all'interno di rete LAN ethernet.
Vengono inviate richieste ARP del tipo `who-has`[1].

Esempio

```
[19:58:19]root@salvador/home/srdjan:# nmap -sn -PR 192.168.1.8

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 19:59 CEST
Nmap scan report for 192.168.1.8
Host is up (0.044s latency).
MAC Address: 00:14:7C:34:2F:D3 (3Com)
Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

Traffico generato

```
# tcpdump -ni em0 host 192.168.1.3
tcpdump: listening on em0, link-type EN10MB
21:08:39.154412 arp who-has 192.168.1.8 (ff:ff:ff:ff:ff:ff) tell 192.168.1.3
21:08:39.154536 arp reply 192.168.1.8 is-at 08:00:27:21:a7:76
```

[1] **EX:** in questo caso funzionava solo nella versione `salvador` → `openpita`

OS detection

Problema

scoprire da remoto il sistema operativo di un host della rete

Perché?

- permette di concentrarsi solo su particolari vulnerabilità
- aiuta nella realizzazione di un exploit
- utili informazioni sulla composizione di una rete
- rilevamento di dispositivi "sospetti"
- social engineering
- ...

OS detection
Come funziona?

```
[20:21:46]root@salvador/tmp:# nmap  -O scanme.nmap.org
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 20:21 CEST
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.18s latency).
Not shown: 995 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
80/tcp   open     http
646/tcp  filtered ldp
9929/tcp open     nping-echo
Device type: WAP|media device|storage-misc|general purpose
Running (JUST GUESSING): Netgear embedded (92%), Western Digital embedded (92%), ...
OS CPE: cpe:/o:linux:kernel:2.6.18 cpe:/o:linux:kernel:2.6 cpe:/o:linksys:linux:2.4
cpe:/o:linux:kernel:2.4 cpe:/o:linux:kernel:2.6.22 cpe:/h:asus:rt-n16 cpe:/o:asus:linux:2.6...
Aggressive OS guesses: Netgear DG834G WAP or Western Digital WD TV media player (92%), ...
No exact OS matches for host (test conditions non-ideal).
Network Distance: 12 hops
OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 304.46 seconds
```

Come?

- invio pacchetti TCP/UDP/ICMP particolari
- sfruttare ambiguità degli standard
- $\Rightarrow$ ogni OS implementa lo stack di rete in modo differente

Port scanning

In cosa consiste?

- attività di "*information gathering*"
- determinare porte **open**, **closed** e **filtered** (i.e., nessuna risposta)

`nmap` <target>

1. <target> convertito in IP tramite DNS
2. ICMP `echo request`, TCK ACK
3. con reverse-DNS IP viene convertito nuovamente nome del host
4. "TCP port scan" delle 1000 porte più comuni
5. output

Port scanning

Perché?

- scoprire possibili target di un attacco
- spesso utilizzato dal malware
- lo *scanning* può riguardare tutte o parte delle porte TCP

Come?

molti metodi diversi! Alcuni esempi:

- `connect()` scan
- `SYN` scan
- `FIN` scan

`connect()` scan

In cosa consiste?

- system call `connect()`
- completamento del 3-way handshake
- una volta completata, la connessione è immediatamente chiusa (`RST`)

Osservazioni

- invasivo
- l'attività di scanning può essere monitorata (log, etc.)
- permette di recuperare informazioni significative (e.g., banner grabbing)

Esempio TCP Connect Scan

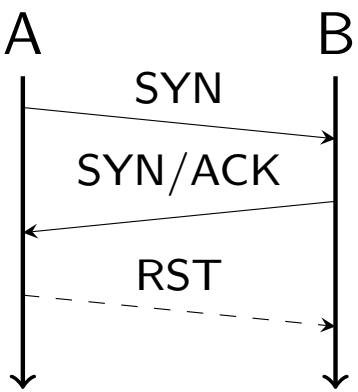
```
[21:33:43]srdjan@gyros~:$ nmap -sT -p22,23,80,5656 kebab

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 21:34 CEST
Nmap scan report for kebab (159.149.148.7)
Host is up (0.00016s latency).
rDNS record for 159.149.148.7: kebab.laser.di.unimi.it
PORT     STATE  SERVICE
22/tcp   open   ssh
23/tcp   closed telnet
80/tcp   closed http
5656/tcp closed unknown

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

SYN scan

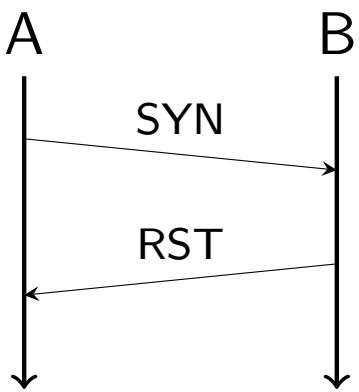
Idea

- perché completare il 3-way handshake?
- . . . mando solo il primo `SYN`!
- 3 possibilità

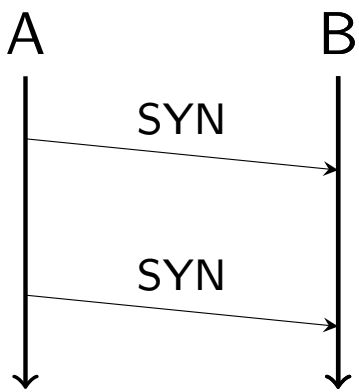
open



closed



filtered



Osservazioni

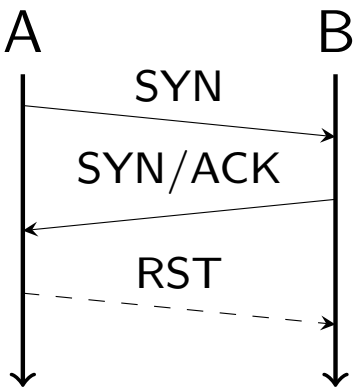
- il 3-way handshake non viene **mai** completato
- $\Rightarrow$ meno invasivo

SYN scan

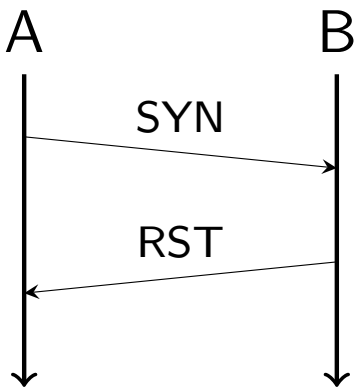
Idea

- perché completare il 3-way handshake?
- . . . mando solo il primo `SYN`!
- 3 possibilità

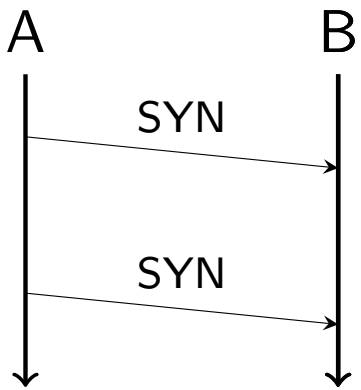
open



closed



filtered



Osservazioni

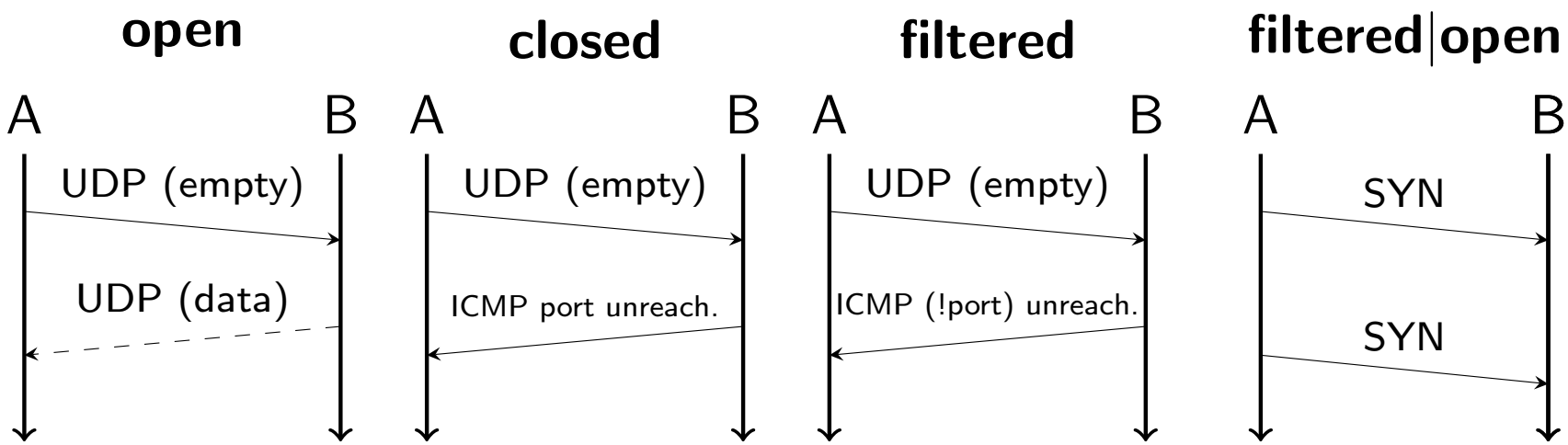
- il 3-way handshake non viene **mai** completato
- $\Rightarrow$ meno invasivo

Esempio TCP SYN (Stealth) Scan:

```
[21:34:18]srdjan@gyros~:$ nmap -sS -p22,23,80,5656 kebab
You requested a scan type which requires root privileges.
QUITTING!
[21:37:15]srdjan@gyros~:$ su
Password:
root@gyros:/home/srdjan# nmap -sS -p22,23,80,5656 kebab

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 21:37 CEST
Nmap scan report for kebab (159.149.148.7)
Host is up (0.00015s latency).
rDNS record for 159.149.148.7: kebab.laser.di.unimi.it
PORT     STATE  SERVICE
22/tcp   open   ssh
23/tcp   closed telnet
80/tcp   closed http
5656/tcp closed unknown
MAC Address: 00:30:05:13:82:EB (Fujitsu Siemens Computers)

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

UDP scan

- diversi servizi usano UDP (DNS, SNMP, DHCP, . . .)
- UDP *non* è connection-oriented



Osservazioni

- se non riceve risposta, la porta è **open|filtered**
- risultati meno affidabili che con TCP

Esempio UDP Scan

```
root@gyros:/home/srdjan# nmap -n -Pn -sU -p 67-68,4567 kebab

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 22:09 CEST
Nmap scan report for kebab (159.149.148.7)
Host is up (0.00015s latency).
PORT     STATE         SERVICE
67/udp   closed        dhcps
68/udp   open|filtered dhcpc
4567/udp closed        unknown
MAC Address: 00:30:05:22:54:AC (Fujitsu Siemens Computers)

Nmap done: 1 IP address (1 host up) scanned in 1.30 seconds
```

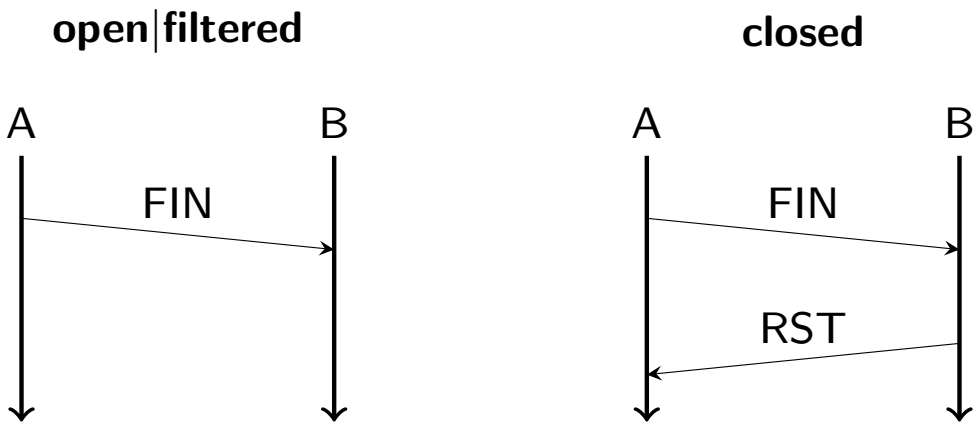
Traffico generato

```
root@kebab:/home/sergio# tcpdump -i eth1 udp
22:09:36.628757 IP gyros.laser.di.unimi.it.57480 > kebab.laser.di.unimi.it.bootpc:  [|bootp]
22:09:36.628784 IP gyros.laser.di.unimi.it.57480 > kebab.laser.di.unimi.it.bootps:  [|bootp]
22:09:36.628839 IP gyros.laser.di.unimi.it.57480 > kebab.laser.di.unimi.it.4567: [|wb]
```

FIN scan

RFC 793, pagina 65-66:

.... "if the [destination] port state is CLOSED... an incoming segment not containing RST causes a RST to be sent in response"
Inoltre in caso di pacchetti inviati a porte "OPEN", privi di bit SYN, RST o ACK
"you are unilickely to get here, but if you do, drop the segment and return"



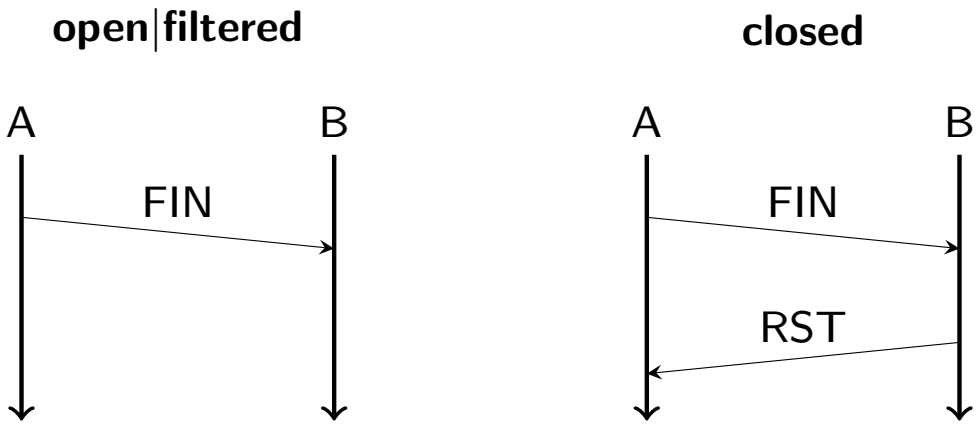
Osservazioni

- ancora meno probabile essere monitorati
- alcuni sistemi (e.g., Microsoft) si discostano dallo standard (RST inviato indipendentemente da porta open o closed)

FIN scan

RFC 793, pagina 65-66:

.... *"if the [destination] port state is CLOSED... an incoming segment not containing RST causes a RST to be sent in response"*
Inoltre in caso di pacchetti inviati a porte "OPEN", privi di bit SYN, RST o ACK
"you are unilickely to get here, but if you do, drop the segment and return"



Osservazioni

- ancora meno probabile essere monitorati
- alcuni sistemi (e.g., Microsoft) si discostano dallo standard (RST inviato indipendentemente da porta open o closed)

Macchina linux

```
root@gyros:/home/srdjan# nmap -n -Pn -sF -p0-2000 kebab

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 23:03 CEST
Nmap scan report for kebab (159.149.148.7)
Host is up (0.00017s latency).
Not shown: 2000 closed ports
PORT   STATE         SERVICE
22/tcp open|filtered ssh
MAC Address: 00:30:05:22:54:AC (Fujitsu Siemens Computers)
Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
```

Macchina Windows

```
root@gyros:/home/srdjan# nmap -sF windowfinestra

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 23:03 CEST
Nmap scan report for windowfinestra (159.149.148.7)
Host is up (0.00012s latency).
rDNS record for 159.149.148.7: windowfinestra.laser.di.unimi.it
Not shown: 999 open|filtered ports
PORT     STATE  SERVICE
3389/tcp closed ms-wbt-server
MAC Address: 00:04:76:AB:CD:EE (3 Com)
Nmap done: 1 IP address (1 host up) scanned in 4.71 seconds
```

Spoofing

Di cosa si tratta?

- *spoofing*: un host tenta di impersonificare un altro host in una comunicazione
- modifica dell'indirizzo sorgente
- consente attacchi "anonimi" (e.g., SYN flood)

Problema

il destinatario accetterà il pacchetto?

Spoofing

Di cosa si tratta?

- *spoofing*: un host tenta di impersonificare un altro host in una comunicazione
- modifica dell'indirizzo sorgente
- consente attacchi "anonimi" (e.g., SYN flood)

Problema

il destinatario accetterà il pacchetto?

Spoofing

Di cosa si tratta?

- *spoofing*: un host tenta di impersonificare un altro host in una comunicazione
- modifica dell'indirizzo sorgente
- consente attacchi "anonimi" (e.g., SYN flood)

Problema

il destinatario accetterà il pacchetto?

- ICMP/UDP: sufficiente modificare l'IP sorgente
- TCP: 3-way handshake e sliding window complicano le cose

L'**idle scan** è una tipologia di scanning inventata nel 1998 dallo stesso autore di `hping`, antirez.

Requisiti per L'Idle scan

- tre attori: il server vittima, una macchina chiamata "zombie", e la macchina da cui viene effettuato lo scan.
- La macchina "zombie, incrementa in modo sequenziale l'ID (Identification).
- La macchina "zombie non deve generare altro tipo di traffico di rete.
- possibilità di effettuare lo spoofing tra la macchina server e la macchina da cui parte lo scan.

Scenario L'Idle scan

1. La macchina da cui parte lo scan, "pinga" continuamente la macchina "zombie", e vede nei pacchetti di ritorno la risposta con id incrementato di 1.
2. La macchina scan invia un pacchetto "spoofato", indirizzo sorgente della macchina "zombie", TCP con SYN attivato su una determinata porta all'host vittima.

Esempio: ping

```
debian:~# hping3 -SA 192.168.1.9 -i 3 -c 3
HPING 192.168.1.9 (eth0 192.168.1.9): SA set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=128 id=2274 sport=0 flags=R seq=0 win=32767 rtt=0.4 ms
len=46 ip=192.168.1.9 ttl=128 id=2275 sport=0 flags=R seq=1 win=32767 rtt=0.2 ms
len=46 ip=192.168.1.9 ttl=128 id=2276 sport=0 flags=R seq=2 win=32767 rtt=0.4 ms
```

Si possono ricever due tipi di risposte:

- se RST la macchina "zombie" non creerà nessun nuovo pacchetto e cosi l'ID rimarrà uguale.
- se SYN/ACK la macchina "zombie", risponderà con un RST e l'ID sarà incrementato di 1.[2]

Esempio: spoofing

```
debian:~# hping3 -S --spoof 192.168.1.9 192.168.1.2 -p 80 -c 1
HPING 192.168.1.2 (eth0 192.168.1.2): S set, 40 headers + 0 data bytes
--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

debian:~# tcpdump -ni eth0
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
00:20:27.733896 IP 192.168.1.9.2680 > 192.168.1.2.80: Flags [S], seq 1215805237, win 512, ...
```

Esempio: ping

```
debian:~# hping3 -SA 192.168.1.9 -c 1
HPING 192.168.1.9 (eth0 192.168.1.9): SA set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=128 id=2278 sport=0 flags=R seq=0 win=32767 rtt=0.2 ms
```

[2] **EX:** http://nmap.org/book/idlescan.html

Si possono ricever due tipi di risposte:

- se RST la macchina "zombie" non creerà nessun nuovo pacchetto e cosi l'ID rimarrà uguale.
- se SYN/ACK la macchina "zombie", risponderà con un RST e l'ID sarà incrementato di 1.[2]

Esempio: spoofing

```
debian:~# hping3 -S --spoof 192.168.1.9 192.168.1.2 -p 80 -c 1
HPING 192.168.1.2 (eth0 192.168.1.2): S set, 40 headers + 0 data bytes
--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

debian:~# tcpdump -ni eth0
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
00:20:27.733896 IP 192.168.1.9.2680 > 192.168.1.2.80: Flags [S], seq 1215805237, win 512, ...
```

Esempio: ping

```
debian:~# hping3 -SA 192.168.1.9 -c 1
HPING 192.168.1.9 (eth0 192.168.1.9): SA set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=128 id=2278 sport=0 flags=R seq=0 win=32767 rtt=0.2 ms
```

[2] **EX:** http://nmap.org/book/idlescan.html

TCP spoofing

`A` vuole fingersi `T` nella comunicazione con `S`

Blind

1. impedire comunicazione `S` $\rightarrow$ `T` (e.g., tramite SYN flooding)
2. predire *initial sequence number* di `S` (problematico)[3]

Non-blind

più semplice: non devo predire nessun valore dell'header TCP

[3] lettura suggerita: *Silence on the Wire*, Michael Zalewski

TCP spoofing

`A` vuole fingersi `T` nella comunicazione con `S`

Blind

1. impedire comunicazione `S` $\rightarrow$ `T` (e.g., tramite SYN flooding)
2. predire *initial sequence number* di `S` (problematico)[3]

Non-blind

più semplice: non devo predire nessun valore dell'header TCP

[3] lettura suggerita: *Silence on the Wire*, Michael Zalewski

In cosa consiste?

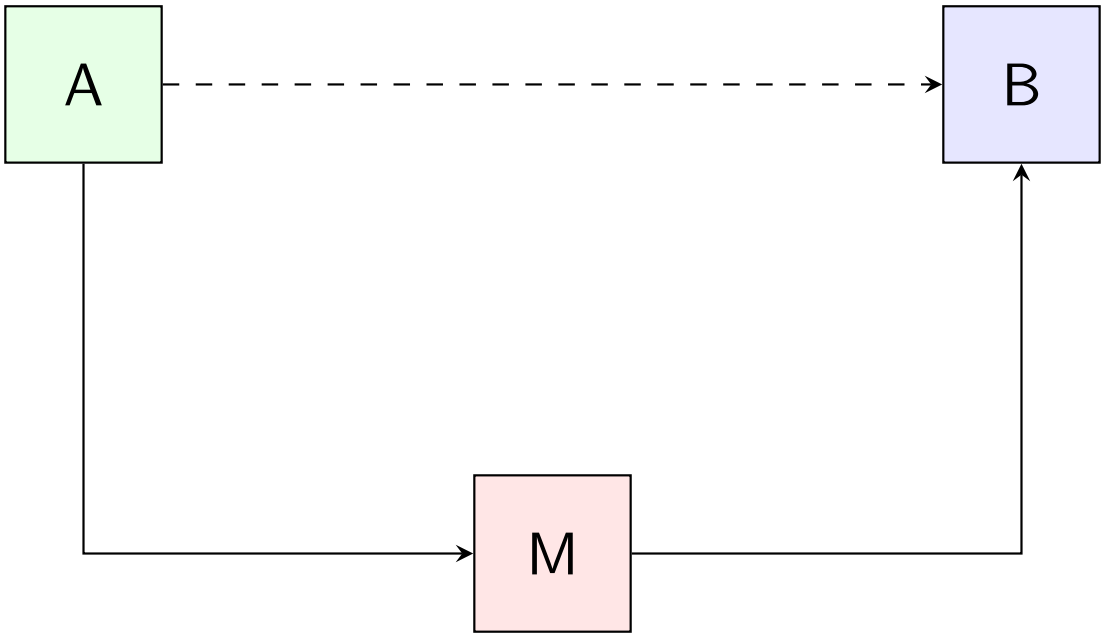
- invio di molti pacchetti TCP SYN
- saturazione della tabella connessioni della vittima
- $\Rightarrow$ DoS (crash o rifiuto altre connessioni)
- *nota*: **pacchetti spoofabili**

Variante: process table flooding

- apertura di molte connessioni TCP
- ogni connessione causa la creazione di un nuovo processo
- $\Rightarrow$ saturazione process table e DoS

4

[4] **EX:** DISEGNO: http://it.wikipedia.org/wiki/SYN_flood



⟶ *connessione fisica*

- - ➛ *connessione logica*

ARP poisoning

Problema

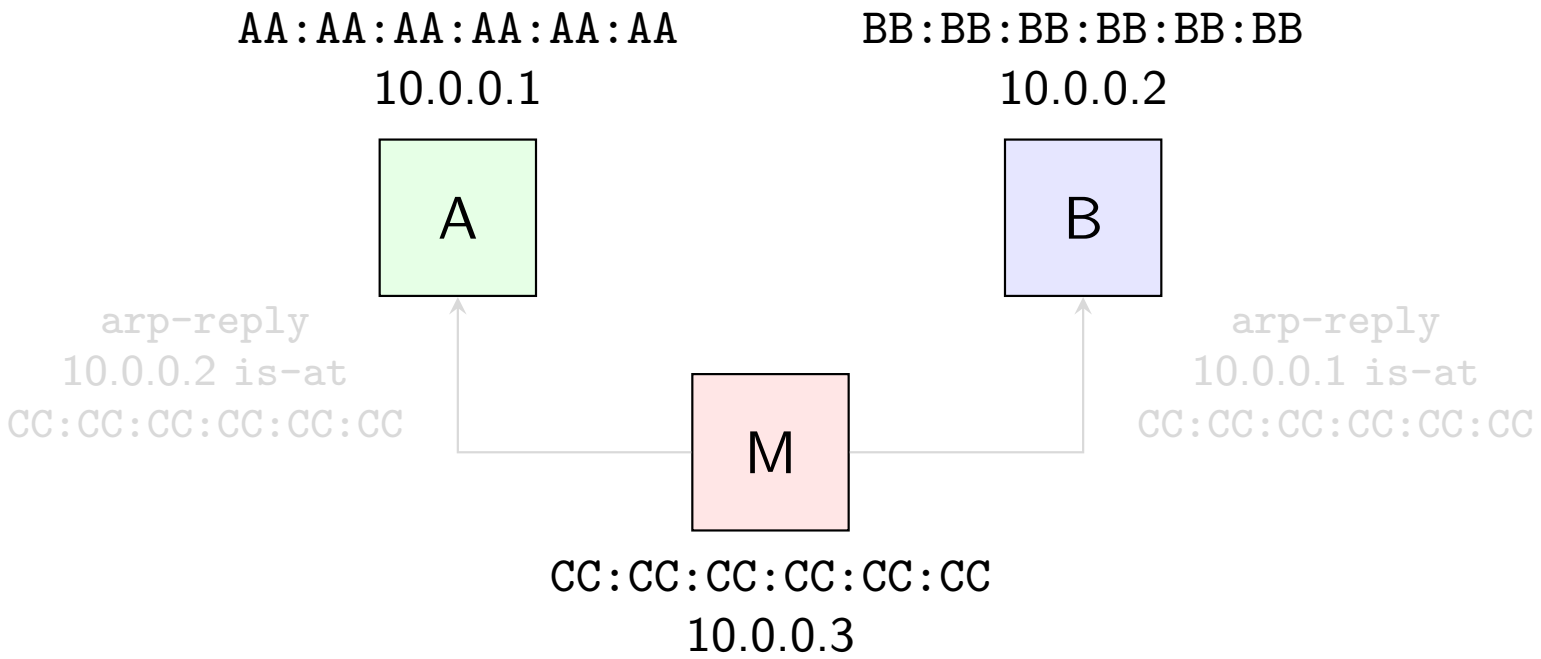
molti sistemi accettano `ARP-reply` per `ARP-request` mai inviate

- ICMP può sostituire gli `arp-reply` (ICMP redirect)

ARP poisoning

Problema

molti sistemi accettano `ARP-reply` per `ARP-request` mai inviate

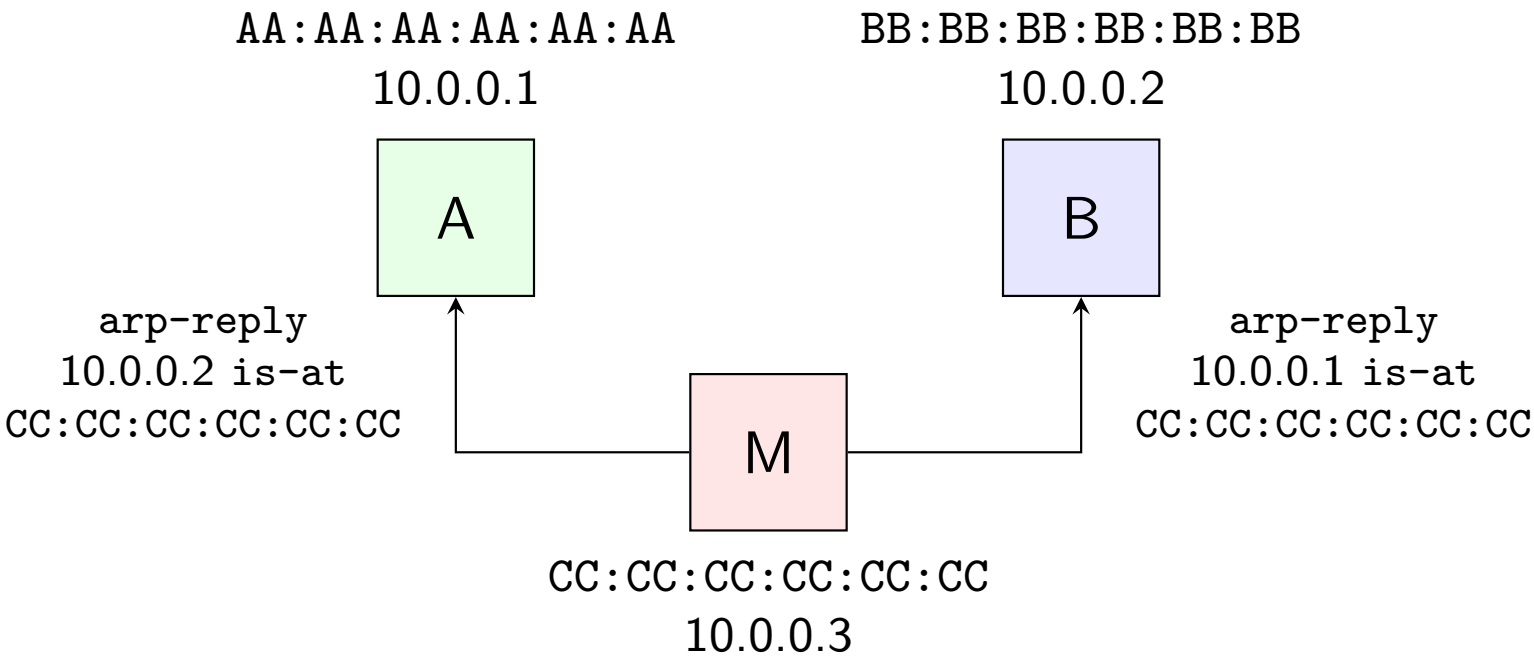


- ICMP può sostituire gli `arp-reply` (ICMP redirect)

ARP poisoning

Problema

molti sistemi accettano `ARP-reply` per `ARP-request` mai inviate

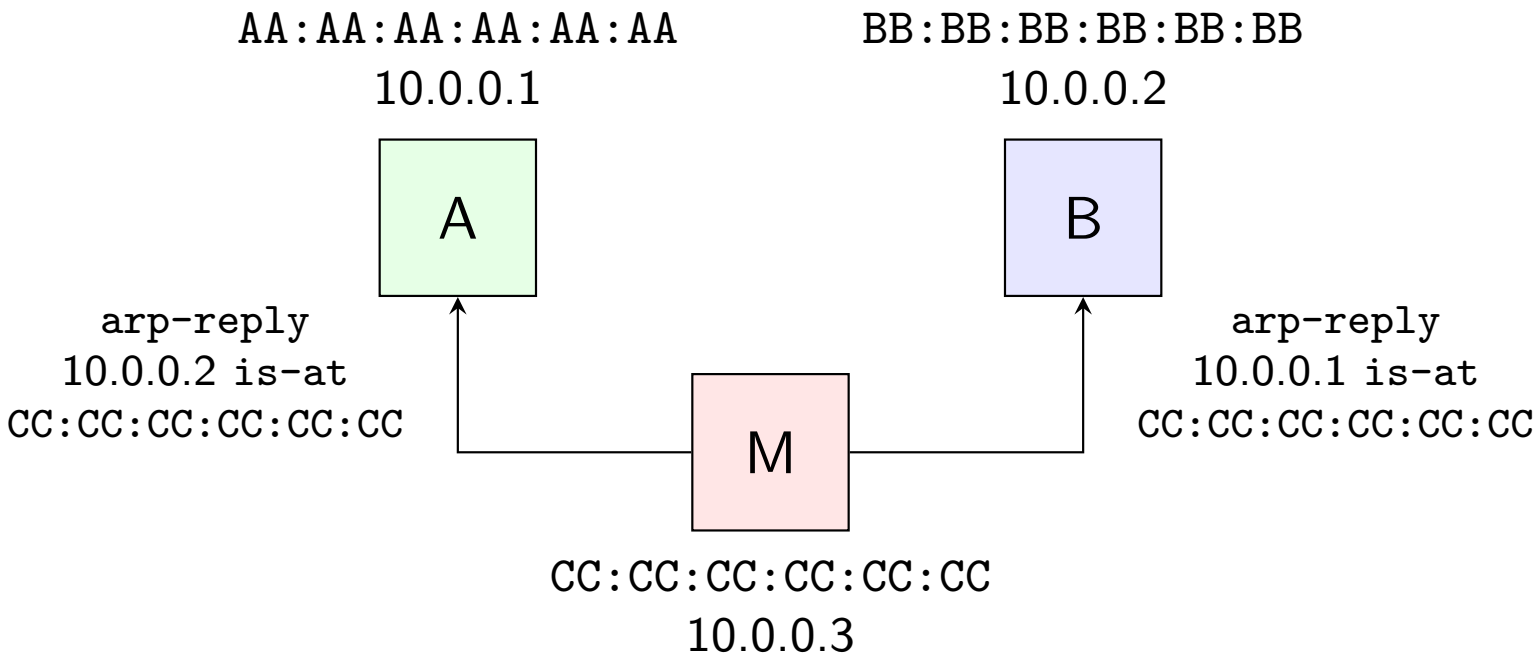


- ICMP può sosituire gli `arp-reply` (ICMP redirect)

ARP poisoning

Problema

molti sistemi accettano `ARP-reply` per `ARP-request` mai inviate



- ICMP può sosituire gli `arp-reply` (ICMP redirect)

Ettercap

Ettercap is a suite for man-in-the-middle attacks on LAN. It features sniffing of live connections, content filtering on the fly and many other interesting tricks.

Esempio

```
$ sudo ettercap -T -M arp:remote /192.168.1.1/ /192.168.1.6/
ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA

Listening on eth1... (Ethernet)
  eth1 ->       00:08:13:34:3B:09       192.168.1.3     255.255.255.0

...

ARP poisoning victims:
 GROUP 1 : 192.168.1.1 00:14:4E:28:A8:E6
 GROUP 2 : 192.168.1.6 00:06:C9:B7:AA:98
```

Esercizio

`Man-page is your best friend!`

0xA0

Enumerare gli host nella propria sottorete, prima con ICMP PING scan, poi con ARP PING scan. Ci sono differenze nei risultati ottenuti con le due modalità? Perché?

Esercizio

0xA1

Effettuare TCP SYN, TCP ACK e UPD Ping sugli host della propria sotto-rete, senza ricorrere al “host discovery”.

Esercizio

0xA2

Effettuare un UDP Ping scan sulle porte 4545, 4646, 4747, 21300-21700, 34000-36181 inviando pacchetti UDP non vuoti.

Esercizio

0xA3

Tramite le VM, creare una mini-rete locate con una macchina Debian, una OpenBSD ed una Windows/MAC. Effettuare lo scan della propria mini-rete di host utilizzando `nmap` con le modalità ICMP ed IP Ping.
Effettuare le stesse prove provando i differenti "timing templates" (opzione "**-T**").

Esercizio

0xB0

Verificare l'accuratezza del sistema di OS detection di `nmap`, cercando di determinare da remoto:

- il sistema operativo di una macchina virtuale (Linux);
- il sistema operativo di una macchina fisica (Windows);
- il sistema operativo utilizzato da un apparato di rete (e.g., router).

Ripetere l'esercizio provando ad utilizzare le opzioni `--ossscan-limit`, `--ossscan-guess`, `--max-os-tries`.

Esercizio

0xC0

Dalla macchina virtuale, fare un port scanning sulla macchina fisica, usando CONNECT, SYN e FIN scan. Che differenze ci sono tra le porte open/closed/filtered riportate con le 3 modalità?

Esercizio

0xC1

Ripetere l'esercizio precedente usando solo SYN scan. Nel frattempo, verificare con `tcpdump` il traffico generato da `nmap`. Corrisponde con quanto atteso? Osservare come `nmap` risponde ad eventuali SYN/ACK ricevuti da porte aperte. Come `nmap` sceglie le porte della macchina remota da analizzare?

Esercizio

0xD0

Sniffare username e password di una sessione `telnet`. Il traffico è in chiaro? Tool suggeriti: `tcpdump`, `ettercap`, `wireshark` etc.

Esercizio

0xD1

Utilizzano il proprio sistema di macchine virtuali all'interno della propria sotto-rete, realizzare effettuare uno "idle scan" utilizzando `nmap`.

Esercizio

0xD2

Utilizzando due macchine (A e B), svolgere le seguenti operazioni:

1. la macchina A ascolta su una socket UDP (utilizzando `netcat`);
2. la macchina B invia ad A pacchetti UDP con mittente spoofato (utilizzare il tool `hping`).

I pacchetti di B vengono accettati correttamente da A? Ripetere lo stesso esperimento utilizzando socket TCP. Che differenze ci sono?

Link e materiale utile:

- *http://www.youtube.com/watch?v=bKUjyeQ78AU, Gordon Fyodor Lyon @DEFCON13*
- *http://www.youtube.com/watch?v=Hk-21p2m8YY, Gordon Fyodor Lyon @DEFCON16*
- *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning, Gordon Fyodor Lyon, Nmap Project (gennaio 2009)*
- *Nmap 6: Network Exploration and Security Auditing Cookbook, Paulino Calderon Pale, Packt Publishing (23 novembre 2012)* ... ma tutto sommato il `man` è più esautivo ed utile.
- *Silence On The Wire: A Field Guide To Passive Reconnaissance And Indirect Attacks, Michal Zalewski, No Starch Pr (30 aprile 2005)*