ADVANCED NETWORK SECURITY

Francesco Palmieri
GARR CERT
fpalmier@unina.it

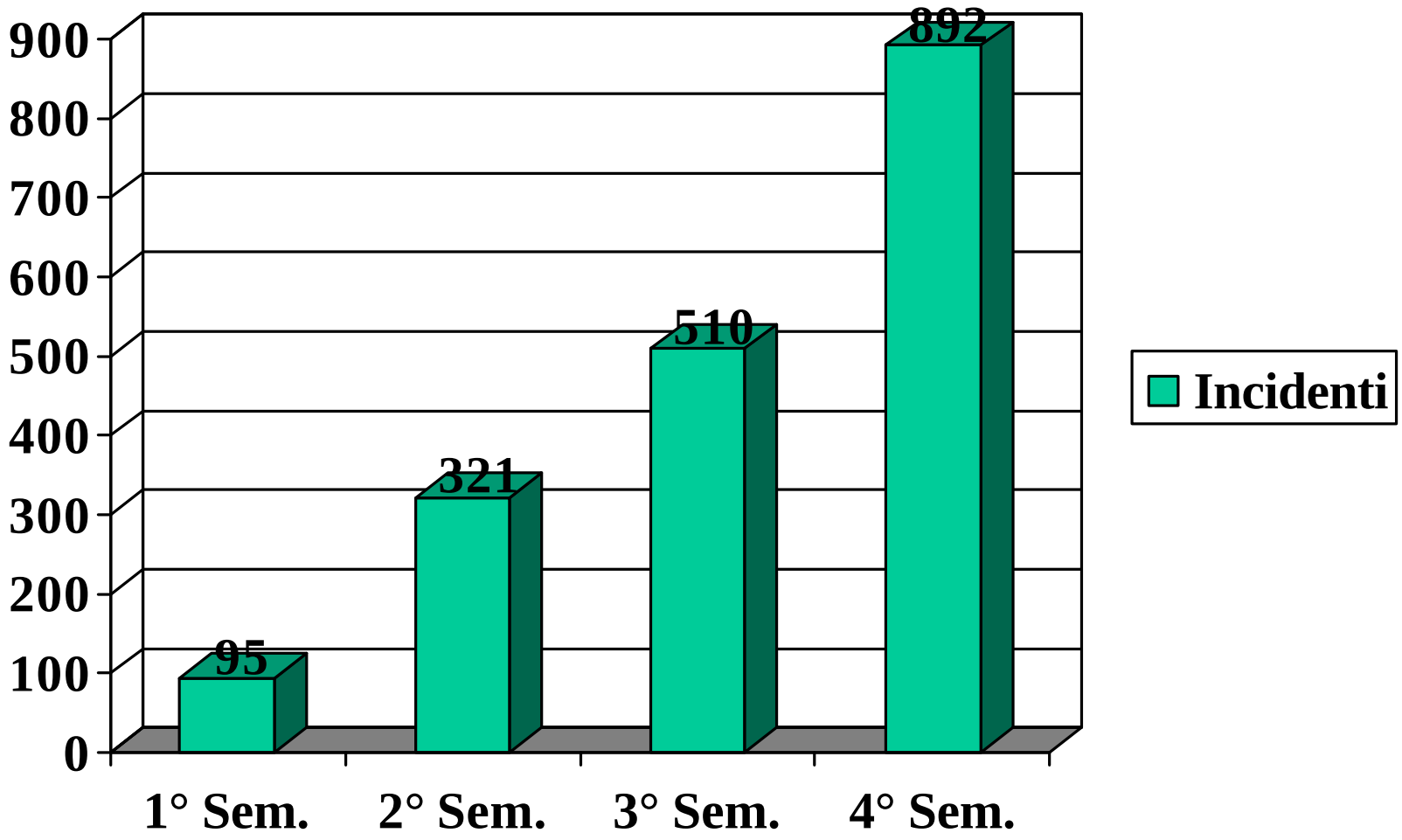
Panoramica degli argomenti

- **Basic Framework**
  - Architettura tipo per una rete sicura
  - Sniffers
  - ACL per il packet filtering
- **Packet Filtering**
  - Tecniche di difesa e protezione
  - Filtraggio dei servizi di rete
  - Anti-spoofing
  - Protezione dell’instradamento
- **Network Scanning**
  - Network Mapping
  - Port Scanning
- **Denial of Service**
  - Tassonomia delle principali tecniche di attacc

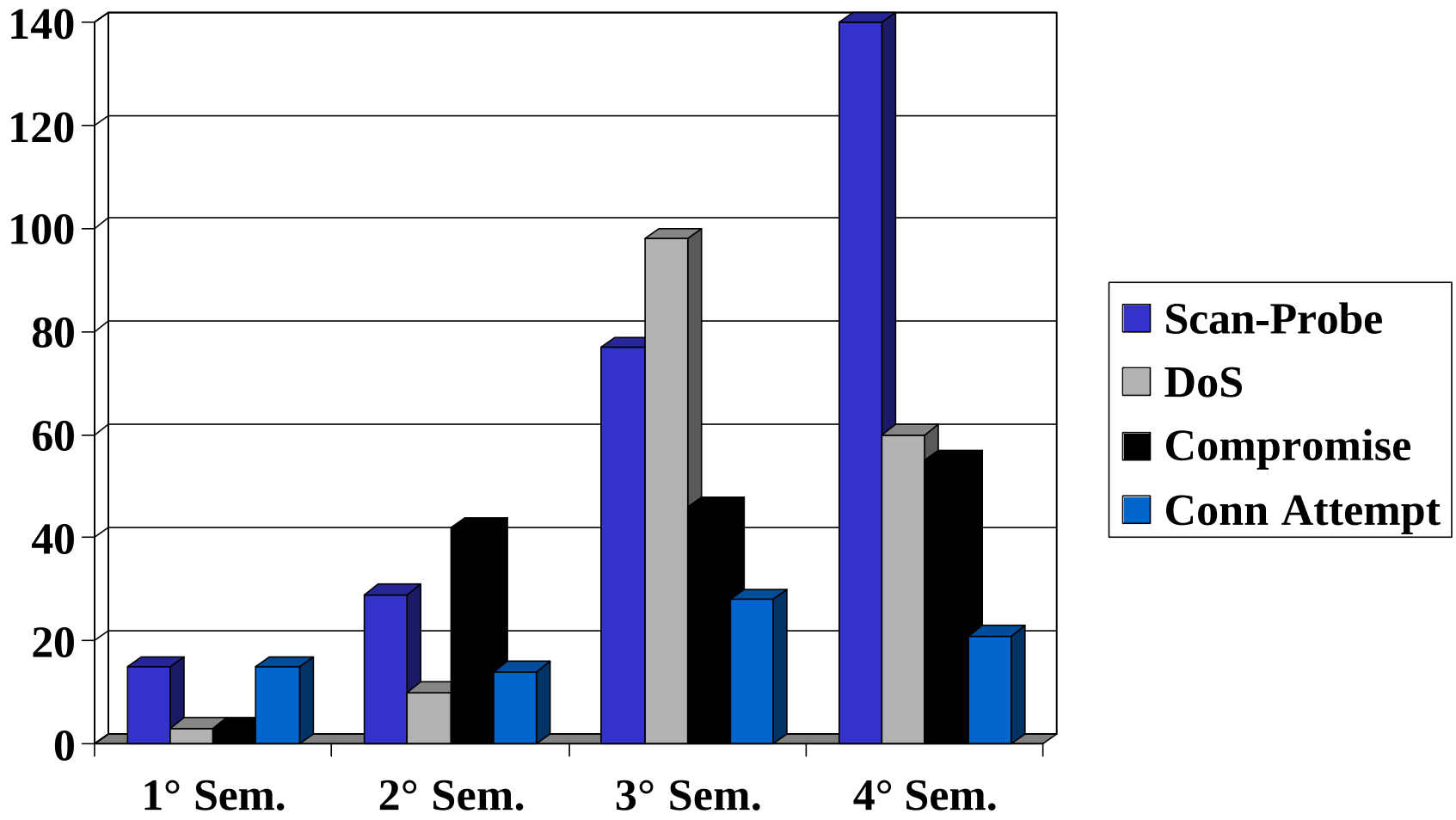
OBIETTIVO

ESERCIZIO DI RETI SICURE!

IL PROBLEMA SICUREZZA IN GARR



LE MAGGIORI CRITICITA’

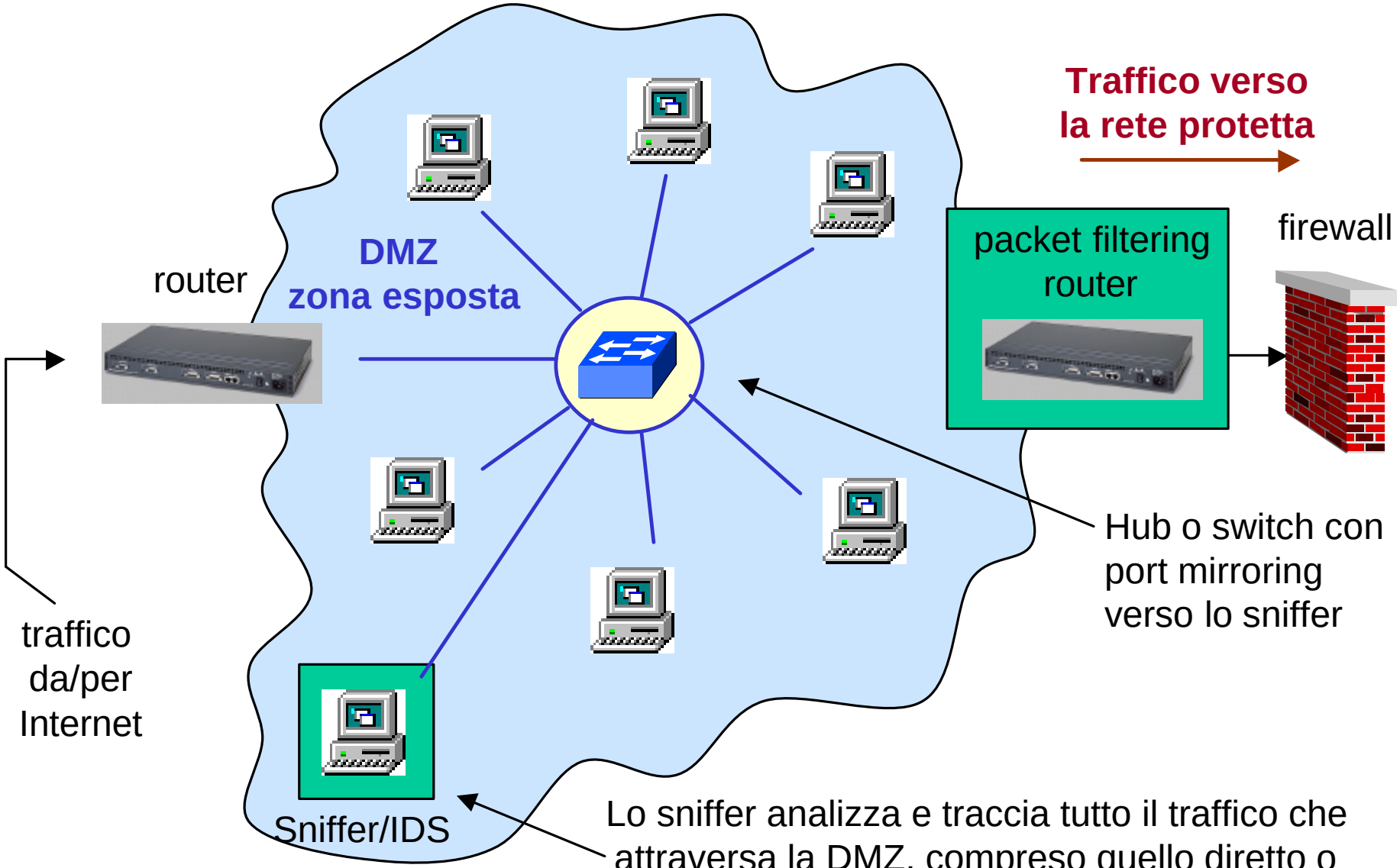


BASIC FRAMEWORK

Gli strumenti di base per l'esercizio di una rete sicura

ARCHITETTURA
SNIFFER
ACCESS LISTS

Architettura di una rete sicura



Packet Sniffing

Sniffer: Strumento software o hardware che sfruttando il promiscuous mode cattura e consente l'analisi di tutti i pacchetti che attraversano un segmento di rete

`tcpdump` : Sniffer public domain basato su Berkeley packet filter (BPF)

Disponibile per il download: **`ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`**

```
23:06:37 10.1.101.1 > 224.0.0.10: ip-proto-88 40 [tos 0xc0]
```

time	source IP	dest IP	protocol	data bytes	type of service

ftp://ftp.isi.edu/in-notes/iana/assignments/protocol-numbers =>

ip-proto-88 = EIGRP

Espressioni di filtraggio BPF

```
08:08:16.155 spoofed.target.net.7 > 172.31.203.17.chargen: udp
```

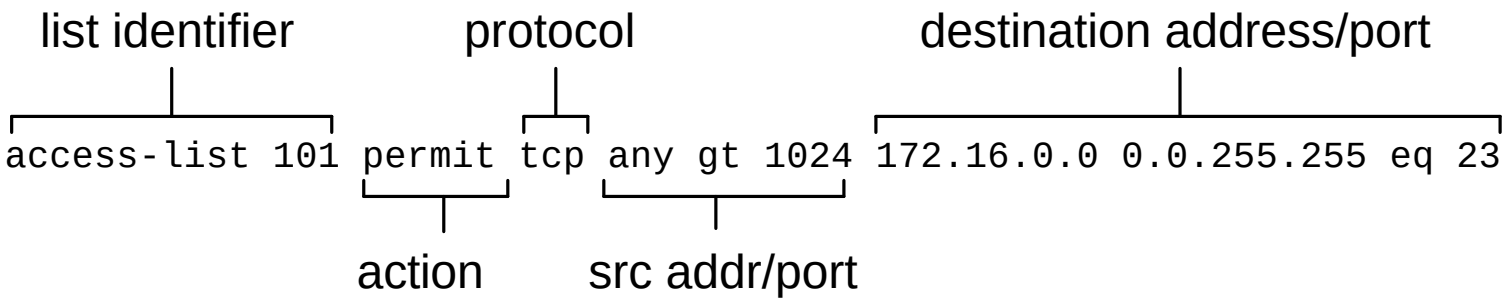
timestamp | src IP | src port | dst IP | dst port | protocol

- gli hosts possono essere referenziati per nome o indirizzo IP
- le porte possono essere specificate per numero o nome del servizio
- per specificare un range di valori vanno indicizzati i bytes specifici

host sorgente o di destinazione:	**`host spoofed.target.net`**
rete di destinazione 172.31.x.x:	**`dst net 172.31`**
reti di destinazione 172.16 - 172.31:	**`dst net 172 and (ip[17]>15) and (ip[17]<32)'`**
porta sorgente 7:	**`src port 7`**
porta destinazione 19:	**`dst port chargen`**
porta sorgente minore di 20:	**`udp[0:2] < 20`**
porta destinazione minre di 20:	**`udp[2:2] < 20`**

ACL - Generalità

- Una ACL è costituita da regole scandite in sequenza fino al primo match



- Ogni ACL termina con una regola “deny any any” implicita
- Non è possibile aggiungere regole nel mezzo di una ACL – E’ necessario distuggere e ricreare l’intero insieme di regole nella sequenza voluta
- A partire dall’ IOS 11.2 si può associare alle ACL un nome logico

```
ip access-list extended allowtelnet permit tcp host 192.132.34.17 any eq 23
```

- La clausola “established” a fine regola identifica tutte le connessioni TCP che hanno superato la fase di setup (3 way handshake)

```
access-list 101 permit tcp any any estabilished
```

ACL - Generalità

- Dalla 12.0 è possibile definire regole in ACL attivabili su base data/ora, specificando un “time-range” di validità e uno scope periodico o assoluto

```
time-range no-http periodic weekdays 8:00 to 18:00
access-list 101 deny tcp any any eq http time-range no-http
```

- Le ACL riflessive, permettono di effettuare operazioni di filtraggio su base sessione del traffico IP, consentendo di autorizzare il traffico relativo a sessioni originate all’interno della propria rete e di bloccare quelle provenienti dall’esterno, aprendo dinamicamente dei varchi nell’accesso per i pacchetti a ritroso relativi a connessioni iniziate dall’interno.

```
! specifica il timeout di durata per le entry temporanee a 300 sec.
ip reflexive-list timeout 300
! L’ACL tmplist viene popolata in base alle sessioni originate dall’interno
ip access-list extended outfilter
 permit tcp any any reflect tmplist
ip access-list extended infilter
permit eigrp any any
 deny icmp any any
 evaluate tmplist
```

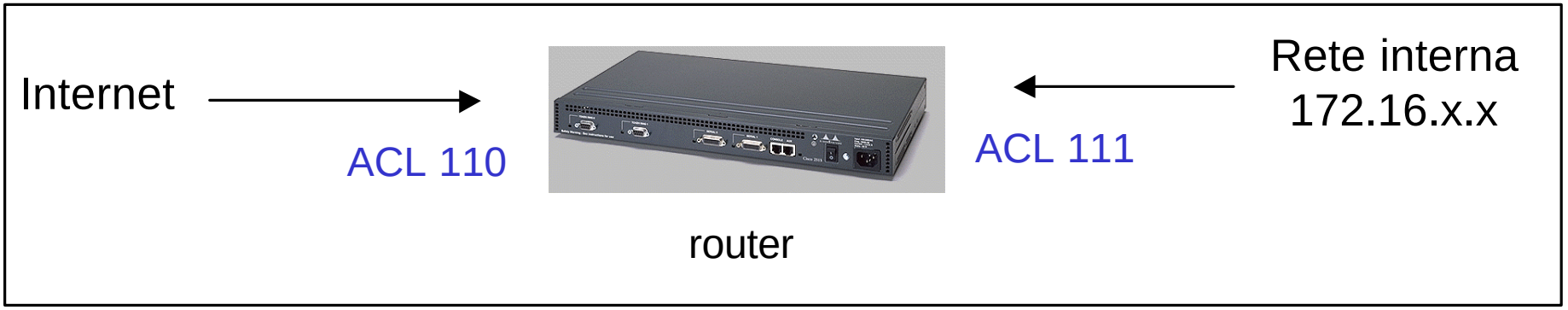
ACL - Generalità

- Una sola ACL può essere applicata alle interfacce di un router in ciascuna specifica direzione (ingresso/uscita):

```
interface Serial0/0
  ip access-group infilter in
  ip access-group outfilter out
```

- E’ possibile filtrare tramite ACL praticamente ogni protocollo instradabile (IP, IPX, Appletalk, NetBIOS source-route bridging, ISO CLNS etc.)

- Negli esempi di seguito le ACL 110 e 111 saranno applicate rispettivamente in ingresso e in uscita sulla sulla border interface che collega al mondo esterno



ACL - Performance

L’uso di ACL complesse con un notevole numero di clausole di filtraggio comporta comunque un certo aggravio prestazionale nell’attività di forwarding dei pacchetti. Per minimizzare l’impatto imposto sulla performance da ACL molto lunghe si consiglia, ove possibile, di utilizzare tecniche avanzate di ottimizzazione dello switching tipo *CEF* e *Netflow Switching*

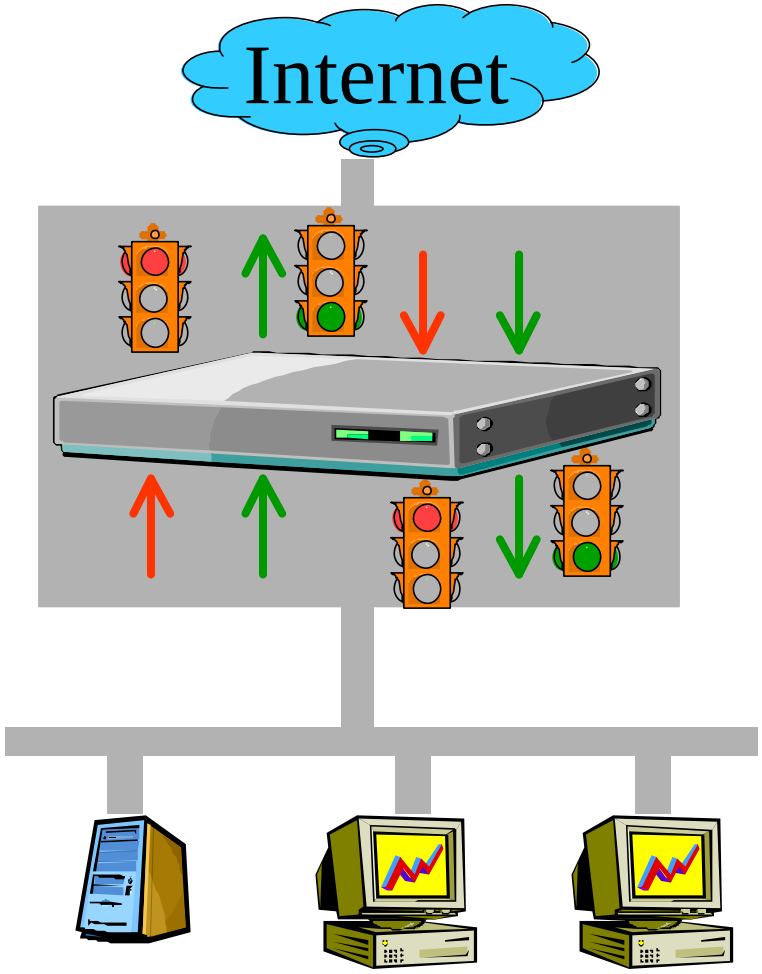
```
! Abilitazione del Distributed CEF
ip cef distributed

! Abilitazione del netflow caching a livello di interfaccia
interface serial6/0/0
    ip route-cache flow
```

PACKET FILTERING

Tecniche di difesa e protezione
Filtraggio dei servizi di rete
Anti-spoofing
Protezione dell’instradamento

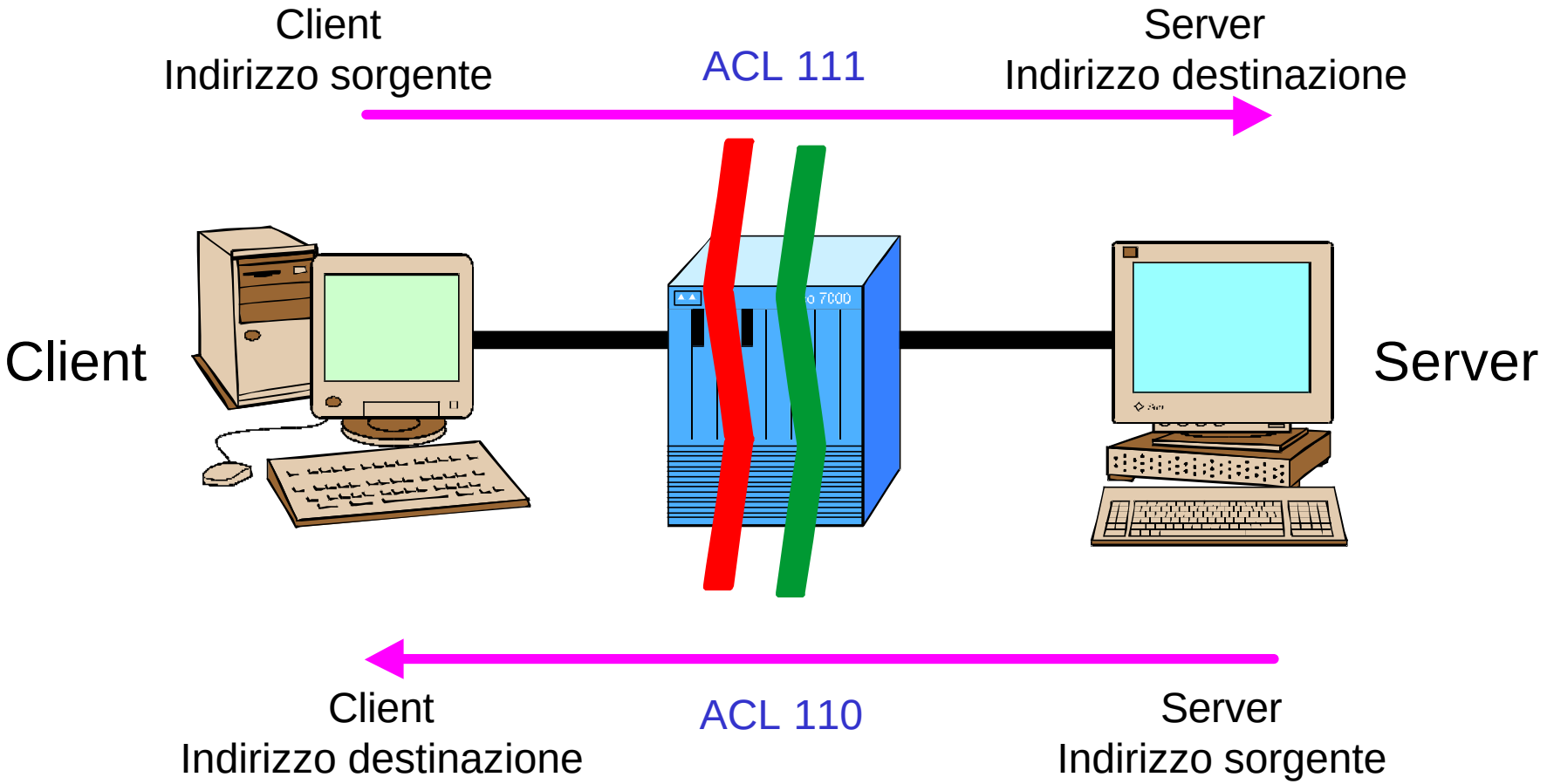
Packet Filtering

- Tecnica di filtraggio del traffico caratterizzata da prestazioni wire-speed e realizzata attraverso uno **Screening Router**
- Realizzabile attraverso l'uso esteso di ACK a livello di:
    - rete e data-link
    - indirizzi IP src e dest
    - numeri porta
    - protocollo (IP, TCP etc.)
    - tipo messaggio ICMP



Filtraggio del traffico – Applicazione ACL

Il filtraggio tramite ACL è praticabile a livello di qualsiasi router della rete ma è più efficace e vantaggioso agire a livello dei router di confine, o "*border router*", che separano autorità e domini di amministrazione distinti e sui cui è possibile controllare in maniera centralizzata i flussi di traffico che attraversano tali domini.



Filtraggio dei servizi

E’ consigliabile bloccare oppure filtrare selettivamente a livello di border router i servizi tendenzialmente pericolosi

Service	Port	Protocol
echo	7	TCP/UDP
discard	9	TCP/UDP
systat	11	TCP/UDP
daytime	13	TCP/UDP
netstat	15	TCP
quotd	17	TCP/UDP
chargen	19	TCP/UDP
ftp-data	20	TCP
ftp	21	TCP
ssh	22	TCP/UDP
telnet	23	TCP
smtp	25	TCP
time	37	TCP/UDP
rlp	39	TCP/UDP
whois	43	TCP/UDP
tacacs	49	TCP/UDP
domain	53	TCP
whois++	63	TCP/UDP
bootp	67-68	UDP
tftp	69	UDP
gopher	70	TCP
finger	79	TCP
http	80	TCP
link	87	TCP
supdup	95	TCP
pop2	109	TCP
pop3	110	TCP
sunrpc	111	TCP/UDP
auth	113	TCP/UDP
nntp	119	TCP
ntp	123	TCP/UDP
nbios-ns	137	TCP/UDP

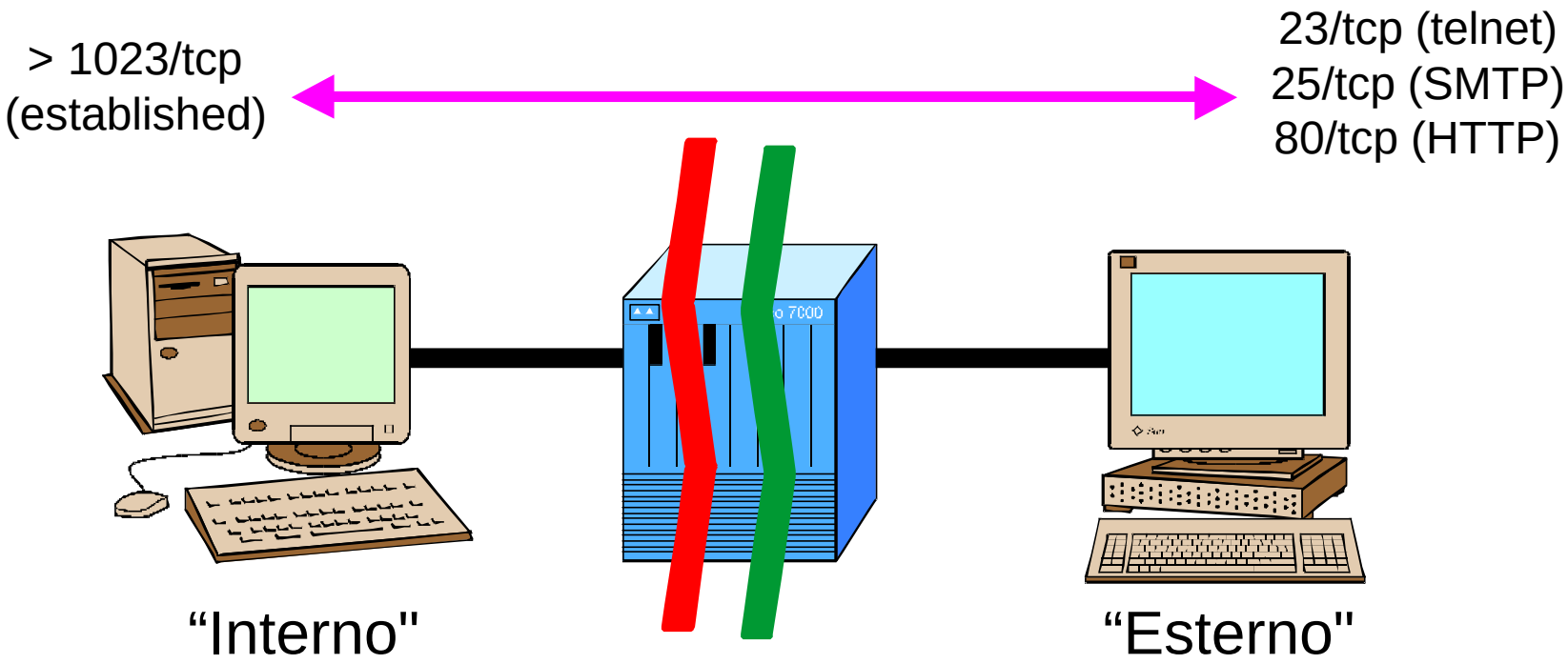
Service	Port	Protocol
nbios-dgm	138	TCP/UDP
nbios-ssn	139	TCP/UDP
imap	143	TCP
NeWS	144	TCP
snmp	161	UDP
snmptrap	162	UDP
xdmcp	177	UDP
irc	194	TCP/UDP
wais/Z39.50	210	TCP
imap3	220	TCP
ldap	389	TCP/UDP
netware-ip	396	TCP/UDP
rmt	411	TCP
https	443	TCP
exec	512	TCP
biff	512	UDP
login	513	TCP
who	513	UDP
shell	514	TCP
syslog	514	UDP
printer	515	TCP/UDP
talk/ntalk	517-518	TCP/UDP
route	520	UDP
timed	525	TCP/UDP
uucp	540-541	TCP
mountd	635	TCP/UDP
wins	1512	TCP/UDP
radius-old	1645-1646	UDP
radius	1812-1813	UDP
openwin	2000	TCP
NFS	2049	TCP/UDP
X11	6000-6063	TCP

Definizione Politiche di Filtraggio

Esempio

- Consenti **in uscita** la fruizione di tutti i servizi (www, e-mail, telnet, news etc.)
- Permetti in ingresso **solo** l'accesso ad un numero estremamente limitato di servizi (e-mail, www, news) erogati solo da hosts specifici e controllati
- Consenti l'FTP in uscita (ed eventualmente in ingresso)
- Consenti Ping e Traceroute dall'interno e dall'esterno
- Consenti in maniera controllata i meccanismi DNS

Servizi consentiti in uscita



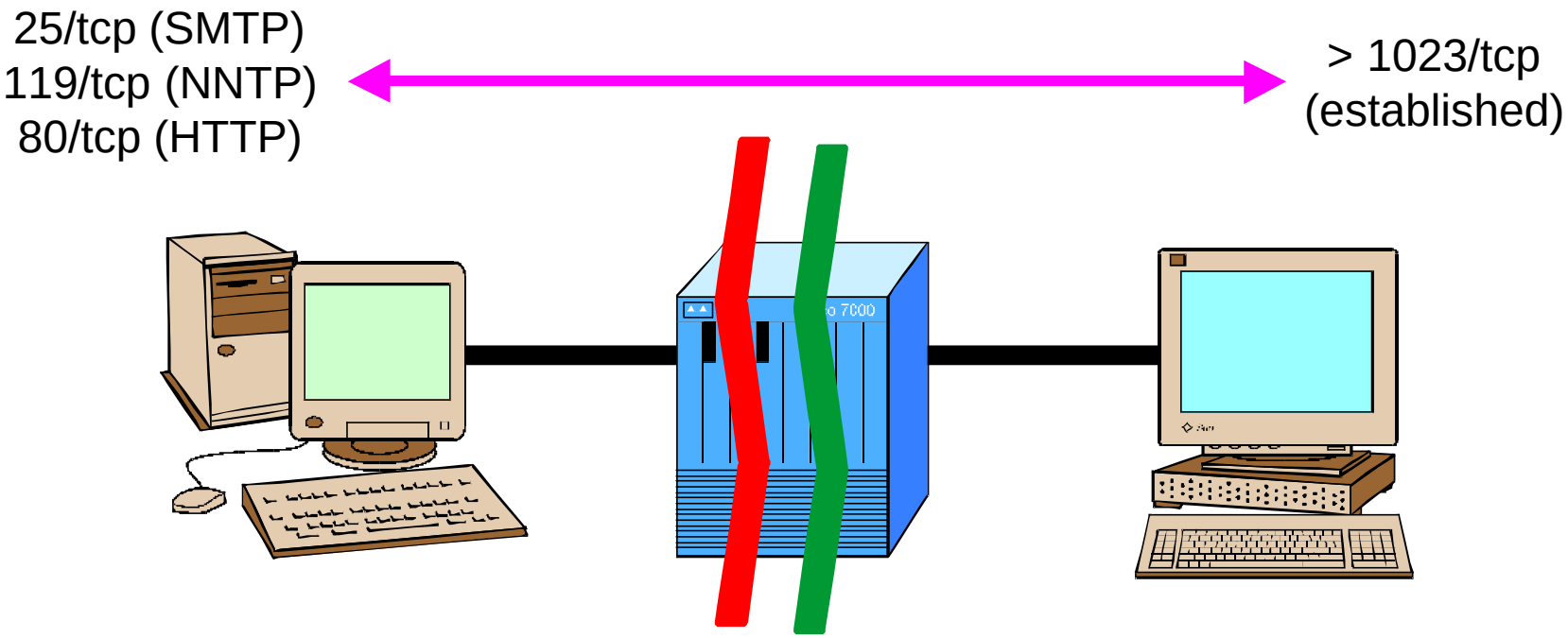
Servizi consentiti in uscita

```
access-list 110 permit tcp any   192.168.1.0 0.0.0.255
   established

access-list 111 permit tcp 192.168.1.0 0.0.0.255 any
```

- Qualsiasi connessione TCP in uscita è consentita senza alcuna restrizione sul servizio

- Sono ammessi in ingresso e a ritroso i soli pacchetti relativi alle sessioni **estabilished,** quindi già aperte dall'interno

Servizi consentiti in ingresso

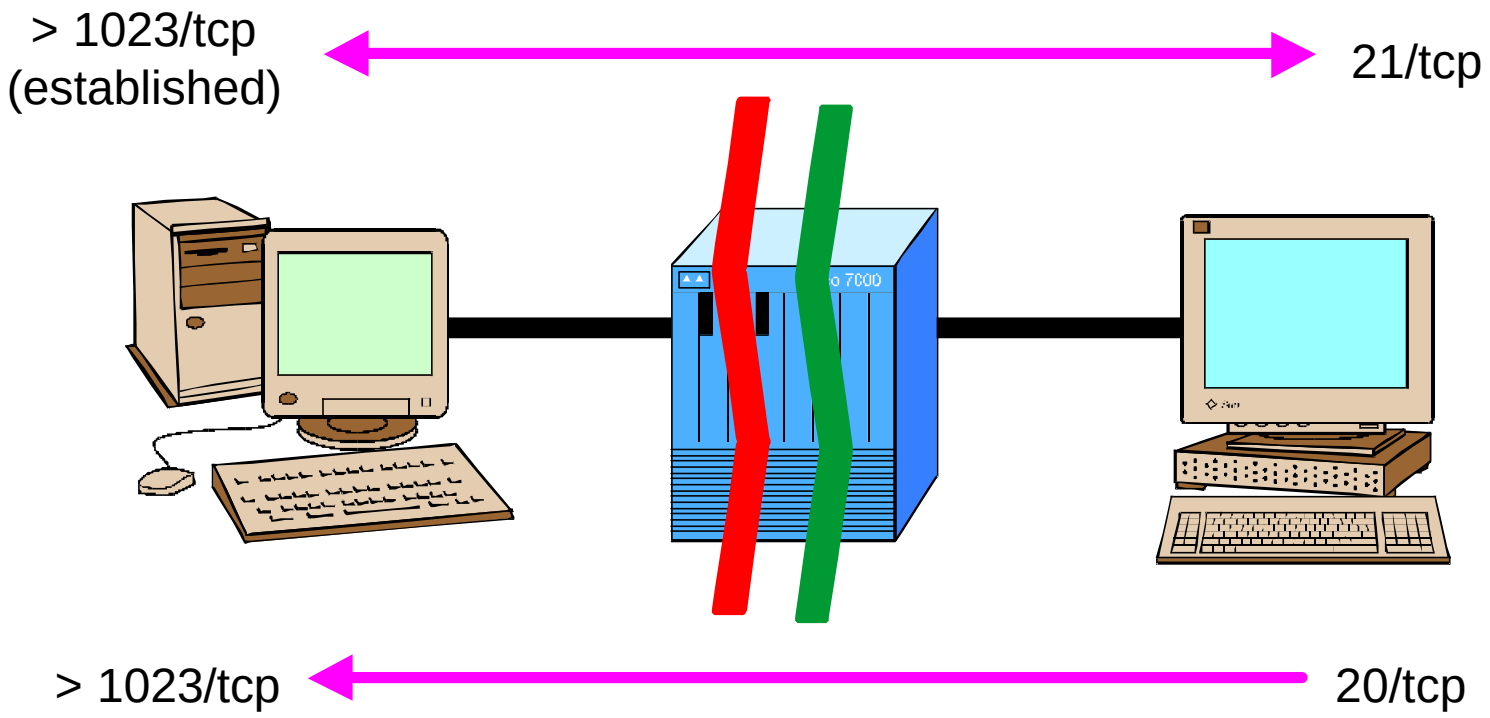


Servizi consentiti in ingresso

```
! Anti Spam
access-list 110 permit tcp any host 192.168.1.1 eq 25
! WWW e News verso i soli servers ufficiali
access-list 110 permit tcp any host 192.168.1.1 eq 80
access-list 110 permit tcp any host 192.168.1.1 eq 119

access-list 111 permit tcp host 192.168.1.1 any established
```

- L’accesso ai servizi interni va controllato con a massima attenzione e consentito solo verso gli hosts erogatori ufficiali di servizi
- Per evitare l’applicazione di esplicite misure anti-spam a livello di tutti gli hosts è conveniente limitare l’accesso SMTP in ingresso ai soli mail-exchangers ufficiali, su cui va concentrata l’applicazione di tutti i meccanismi di protezione (anti-relay etc.)
- La regola di filtraggio in ingresso (ACL 111) è ridondante in quanto già prevista nel controllo del traffico uscente

Il Problema FTP



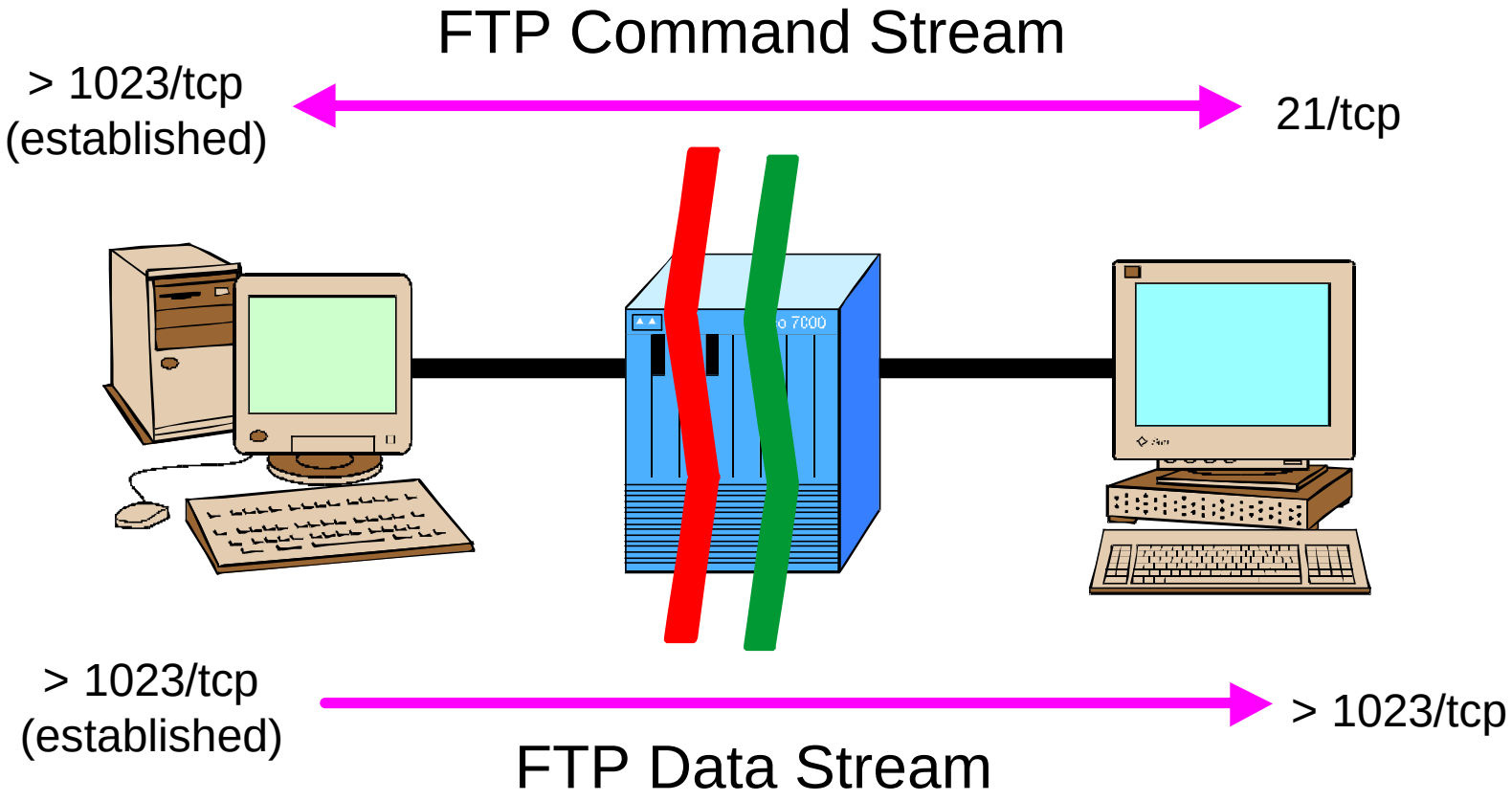
Il Problema FTP

! Consenti le connessioni dati aperte dal server al client sulla porta 20

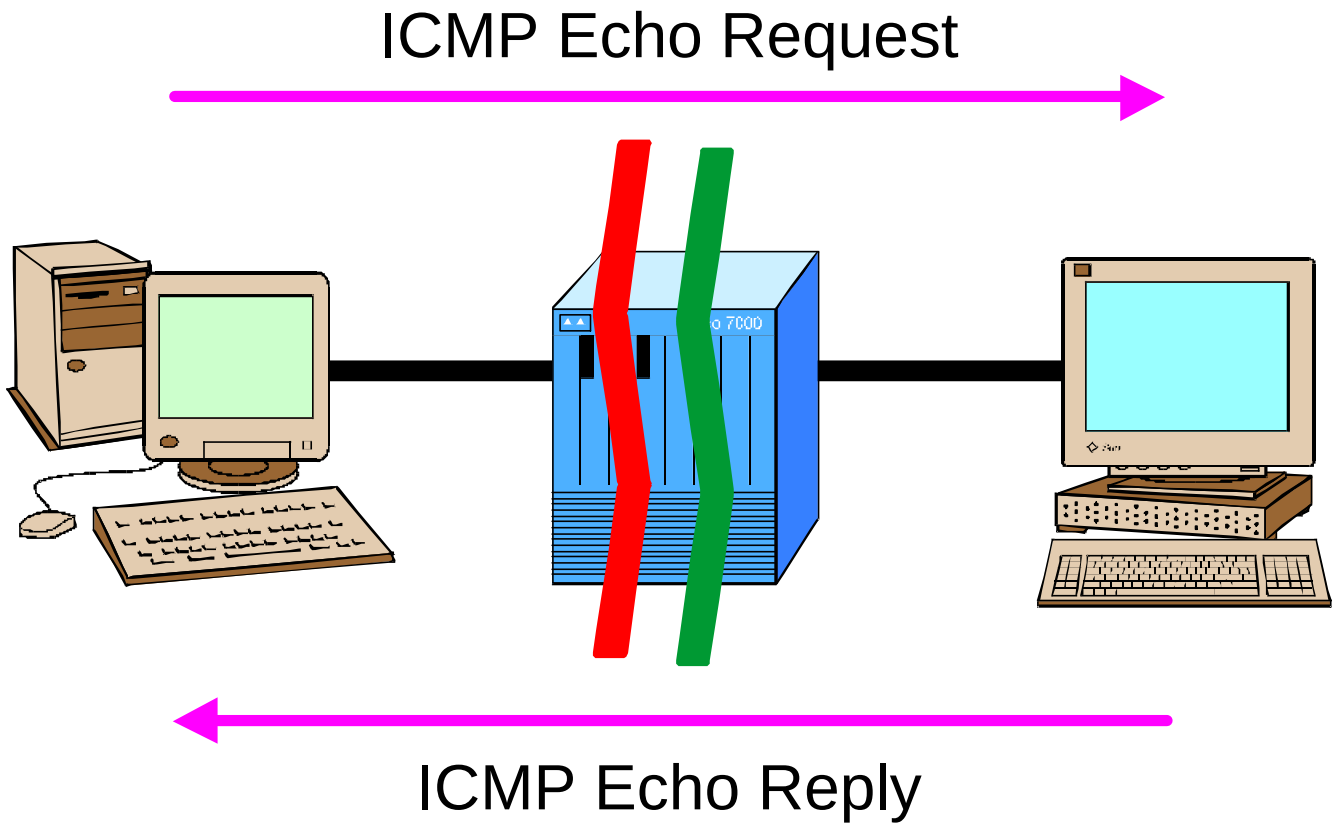
```
access-list 110 permit tcp any eq 20     192.168.1.0
   0.0.0.255 gt 1023

access-list 111 permit tcp 192.168.1.0 0.0.0.255
   any
```

- Dopo aver aperto dall'interno la connessione al canale di contriollo è necessario per ciascun trasferimento garantire la possibilità di aprire a ritroso le connessioni dati
- Lo spoofing della porta sorgente (20) può garantire l'accesso fraudolento a tutte le porte maggiori di 1023
- Per prevenire il probema è meglio usare un firewall"stateful", un proxy server per ftp **oppure consentire l'FTP solo in modalità passiva**

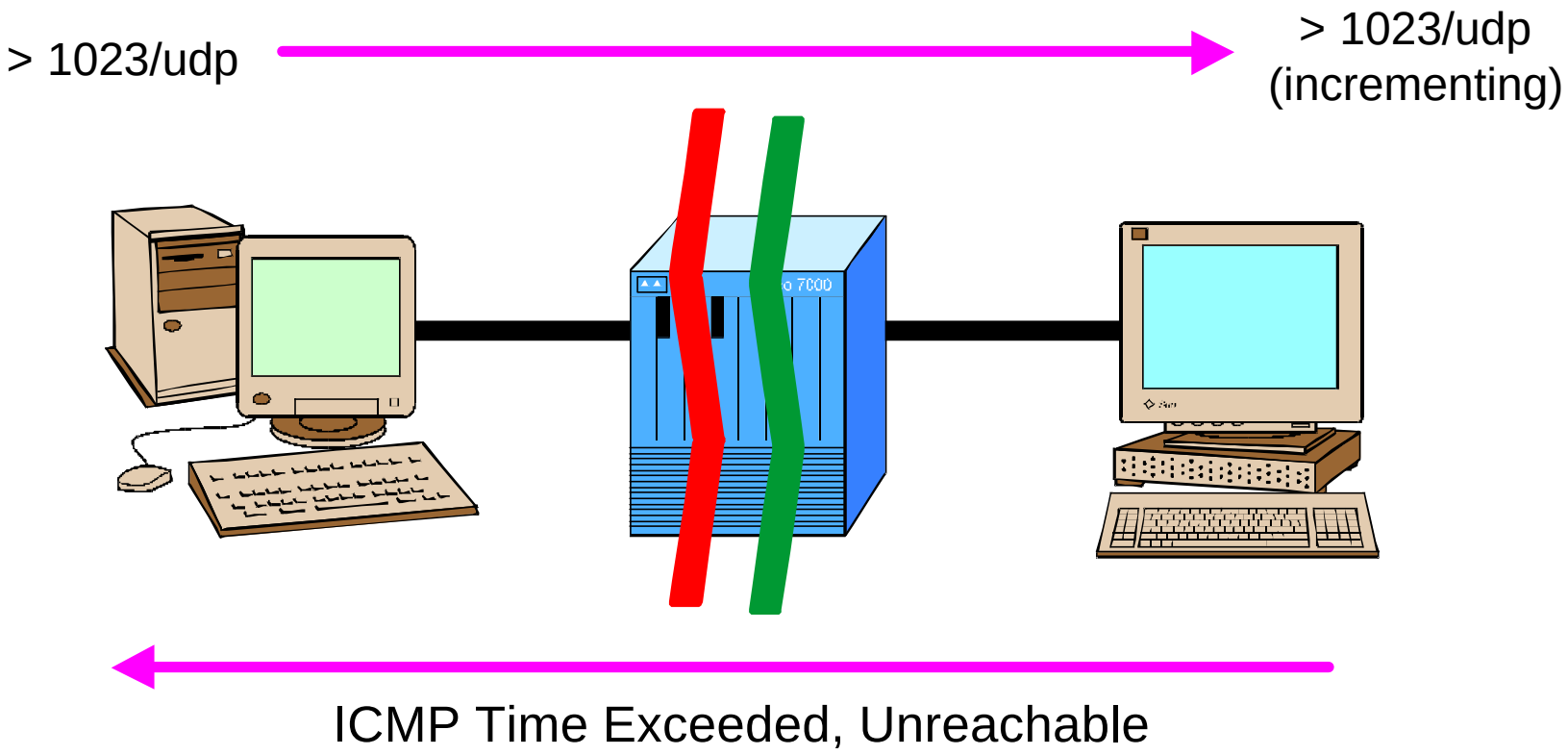
Soluzione: FTP in modalità passiva



Visibilità esterna - Ping



Visibilità esterna - Traceroute

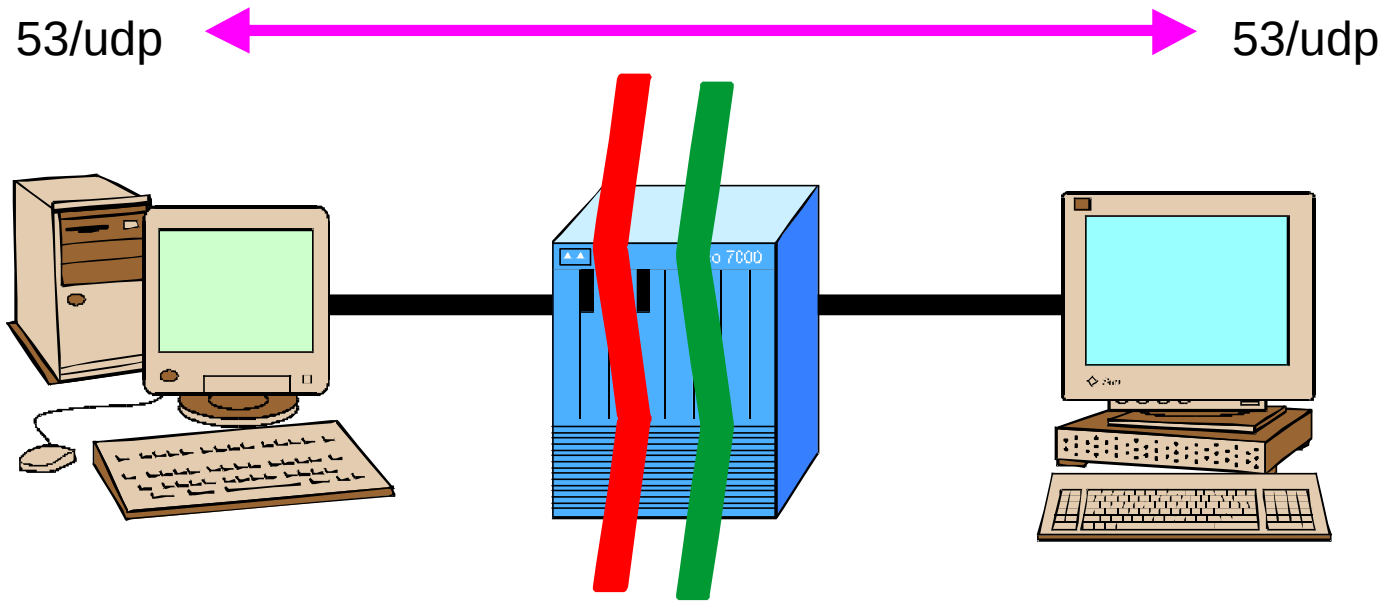


Visibilità - Ping/Traceroute

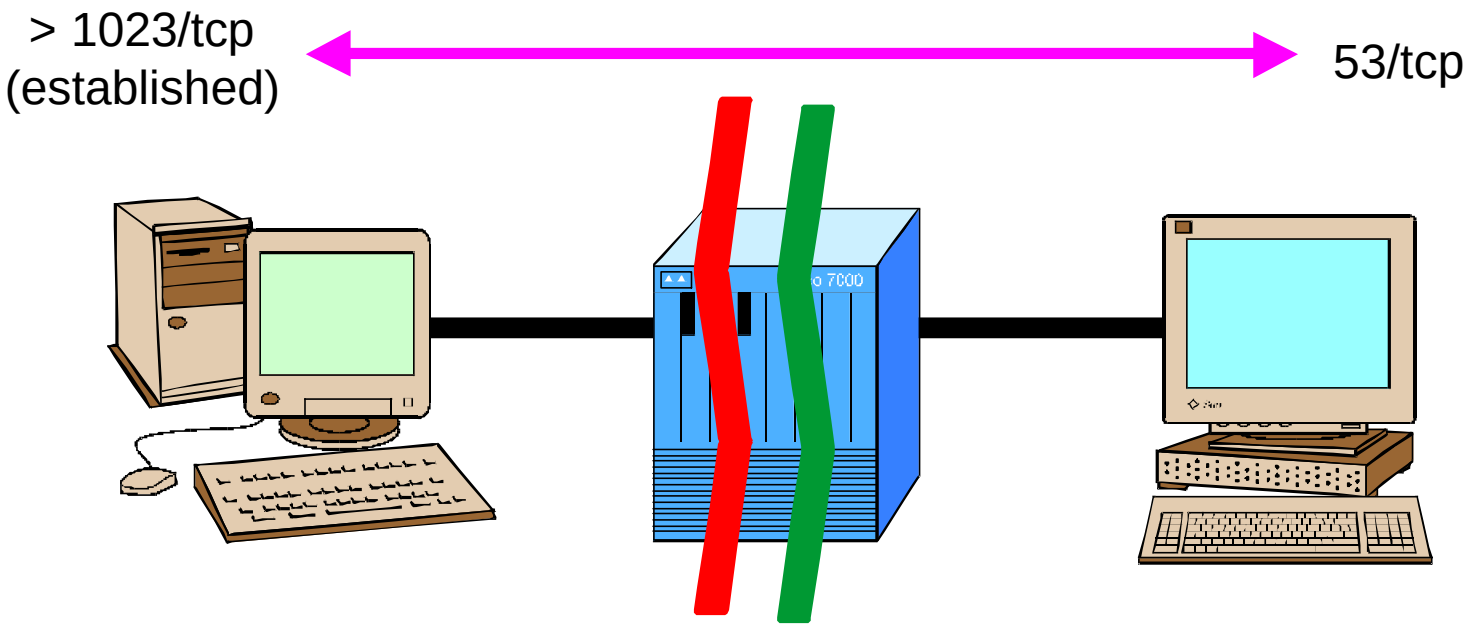
```
access-list 110 permit icmp any 192.168.1.0 0.0.0.255
   echo-reply
access-list 110 permit icmp any 192.168.1.0 0.0.0.255
   time-exceeded
access-list 110 permit icmp any 192.168.1.0 0.0.0.255
   unreachable

access-list 111 permit icmp 192.168.1.0 0.0.0.255
       any echo
access-list 111 permit udp  192.168.1.0 0.0.0.255 gt 1023
        any gt 1023
```

- Consenti solo il traffico ICMP strettamente necessario a garantire la funzionalità dei meccanismi di ping e traceroute

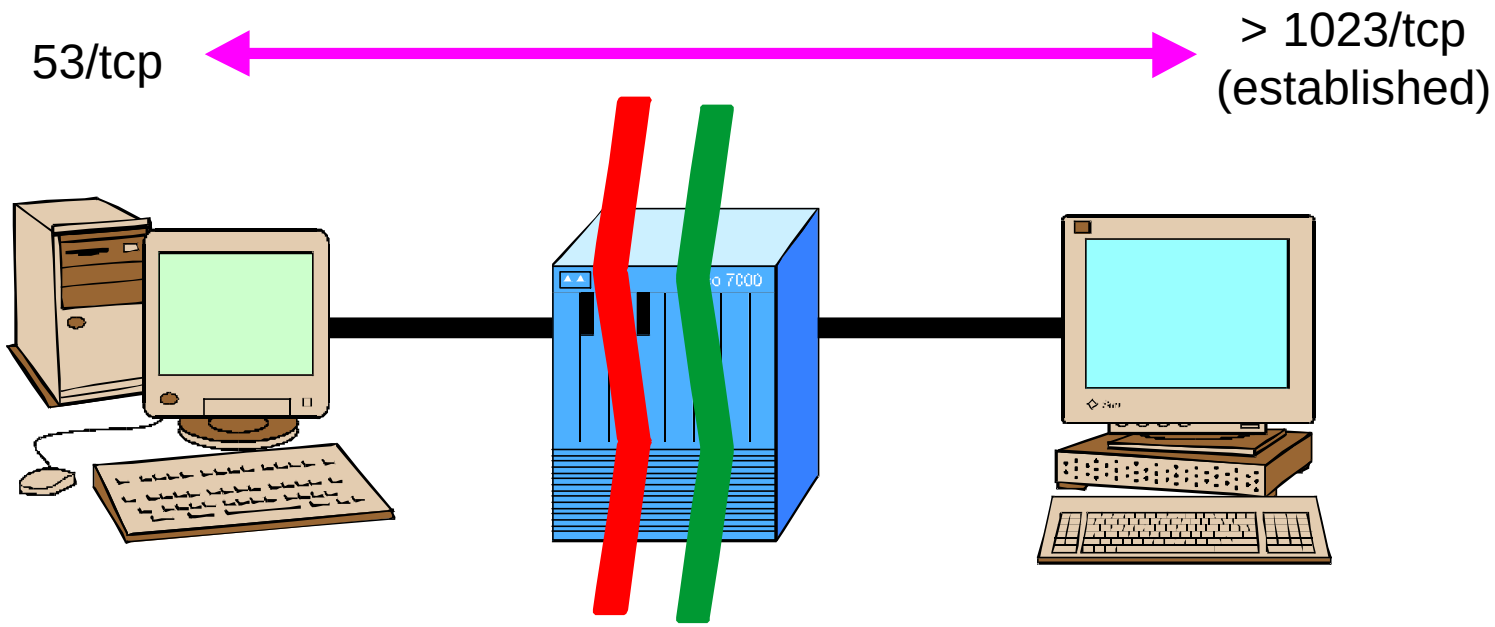
DNS – Attività di Query



DNS – Risposte Bulk



DNS -- Zone Transfers



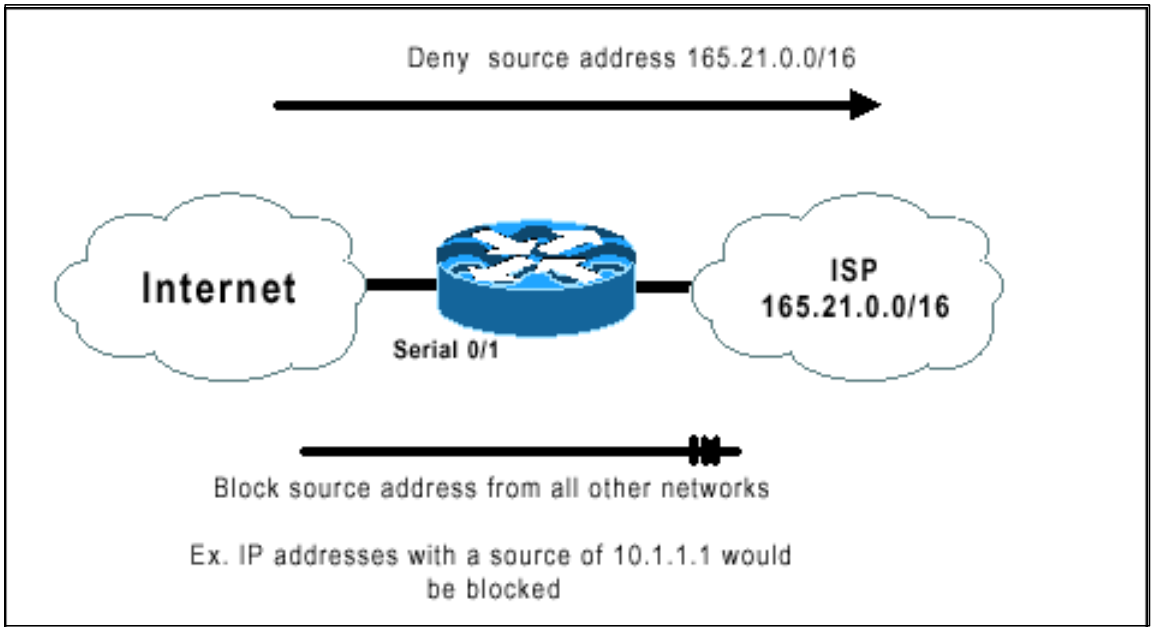
DNS – Regole di filtraggio

```
access-list 110 permit udp any eq 53 host 192.168.1.1 eq 53
access-list 110 permit tcp host 172.16.1.1 host 192.168.1.1 eq 53
access-list 110 permit tcp any host 192.168.1.1 established
!
access-list 111 permit udp host 192.168.1.1 eq 53   any eq 53
access-list 111 permit tcp host 192.168.1.1 gt 1023 any eq 53
access-list 111 permit tcp host 192.168.1.1 eq 53   host 172.16.1.1
   established
```

- Blocca tutte le queries effettuate dall'esterno va TCP
- Consenti i zone transfers con i soli hosts autorizzati e specificamente abilitati a livello di configurazione DNS
- Consenti i trasferimenti "bulk" via TCP solo se iniziati dall'interno

Anti-Spoofing in ingresso

Gran parte degli attacchi in rete, in particolar modo i Denial Of Service si basano sulla falsificazione fraudolenta, o *spoofing*, degli indirizzi d'origine.



Anti-Spoofing in ingresso

Il modo più semplice di proteggersi è quello di scartare tutto il traffico in ingresso con indirizzi sorgente inammissibili rispetto alla provenienza, riservati (RFC 1918) o non correttamente instradabili

! Blocca i traffico dall'esterno con indirizzi sorgente interni:

```
access-list 110 deny ip 165.21.0.0  0.0.255.255   any log
access-list 110 permit ip any any
```

! Blocca i traffico dall'esterno con indirizzi IP non instradabili:

```
access-list 110 deny ip 10.0.0.0    0.255.255.255 any log
access-list 110 deny ip 172.16.0.0  0.15.255.255  any log
access-list 110 deny ip 192.168.0.0 0.0.255.255   any log
access-list 110 deny ip 127.0.0.0   0.255.255.255 any log
access-list 110 deny ip 255.0.0.0   0.255.255.255 any log
access-list 110 deny ip 0.0.0.0     0.255.255.255 any log

interface Serial0/1
       ip access-group 110 in
```

Anti-Spoofing: indirizzi riservati

Reti riservate non instradabili su internet (RFC 1918)

```
10.0.0.0        - 10.255.255.255
172.16.0.0      - 172.31.255.255
192.168.0.0     - 192.168.255.255
```

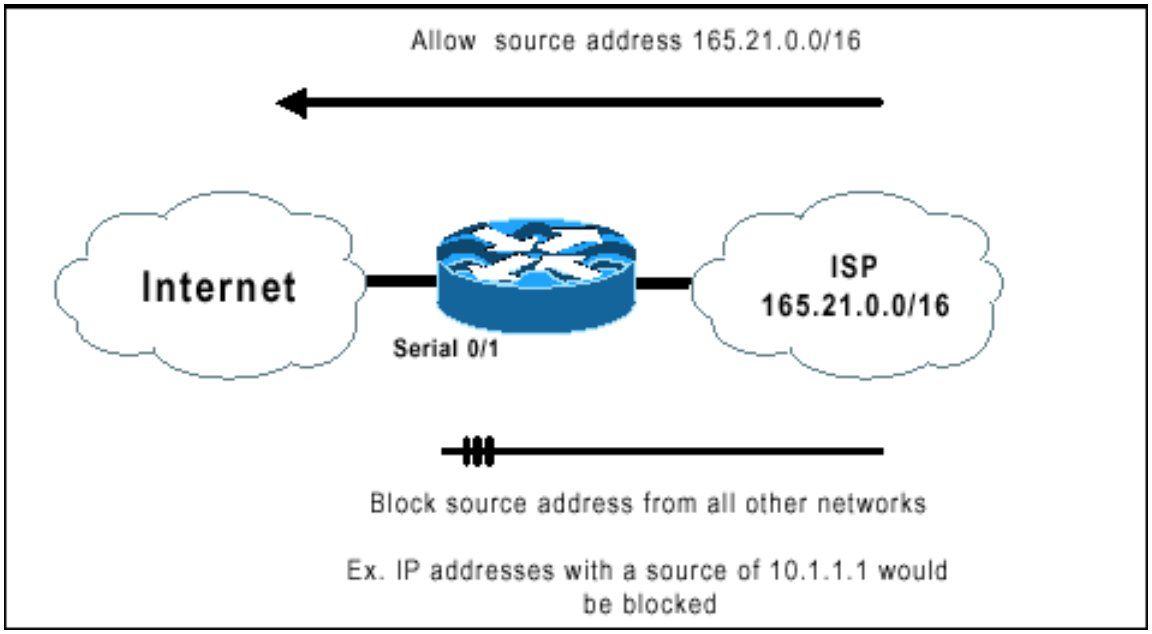
Reti riservate a livello IANA

```
0.0.0.0         - 0.255.255.255
1.0.0.0         - 1.255.255.255
2.0.0.0         - 2.255.255.255
5.0.0.0         - 5.255.255.255
23.0.0.0        - 23.255.255.255
31.0.0.0        - 31.255.255.255
64.0.0.0        - 95.255.255.255
96.0.0.0        - 126.255.255.255
127.0.0.0       - 127.255.255.255
191.255.0.0     - 191.255.255.255
197.0.0.0       - 197.255.255.255
201.0.0.0       - 201.255.255.255
223.255.255.0 - 223.255.255.255
240.0.0.0       - 255.255.255.255
```

Anti-Spoofing in uscita

Per prevenire inoltre spoofing, volontari o involontari, dall'interno della propria rete verso l'esterno, analoghe misure di filtraggio vanno applicate in uscita

Le linee guida per il filtraggio anti-spoofing finora illustrate sono descritte estesamente nella RFC 2267



Anti-Spoofing in uscita

E’ necessario scartare tutto il traffico in uscita caratterizzato da indirizzi sorgente inammissibili, perché non previsti a livello di instradamento, sulla rete di provenienza.

```
! Blocca il traffico uscente con IP sorgente estranei:
access-list 111 permit ip 165.21.0.0 0.0.255.255 any
access-list 111 deny ip any any log


! Applica l’ACL in uscita sulla border interface
interface Serial0/1
       ip access-group 111 out
```

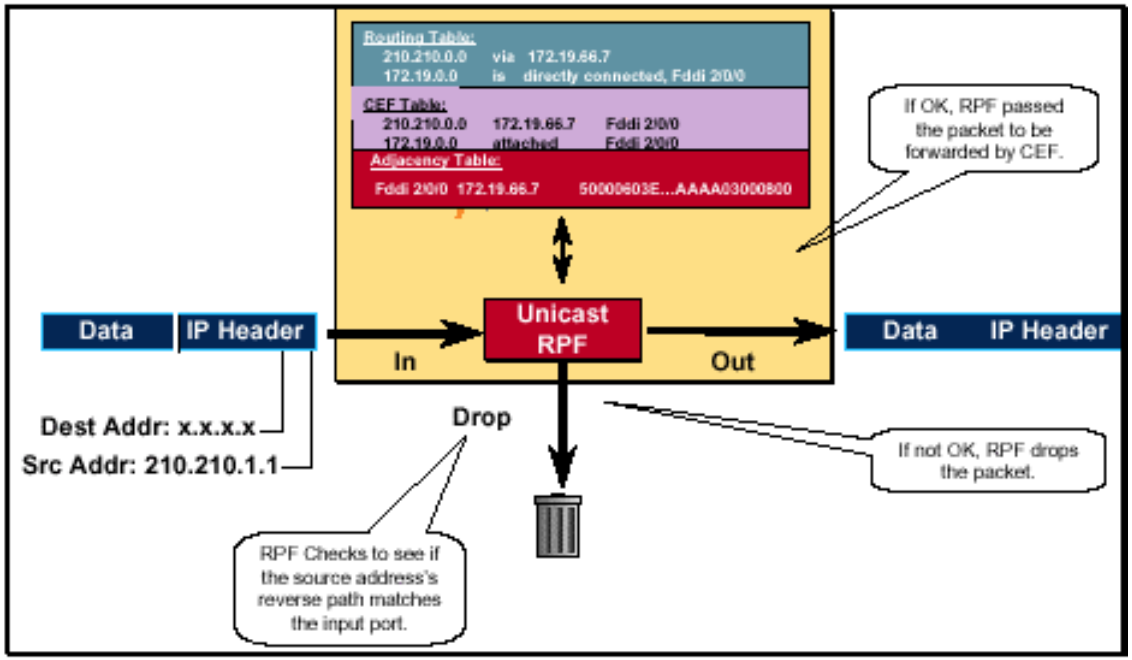
Anti-Spoofing via Unicast RPF

Un modo estremamente elegante ed efficace di inibire lo spoofing è quello di sfruttare la feature di *Unicast Reverse Path Forwarding* (RPF) su versioni IOS basate su *RSP* che supportano la funzionalità di *Cisco Express Forwarding* (CEF). Tale meccanismo è utilizzabile solo in presenza di routing totalmente simmetrico

```
! Abilita il CEF:
ip cef

interface Serial0/1
        ip verify unicast reverse-path

! Su architetture VIP2/4 Abilita il distributed CEF:
ip cef distributed

interface Serial4/0/1
        ip verify unicast reverse-path
```

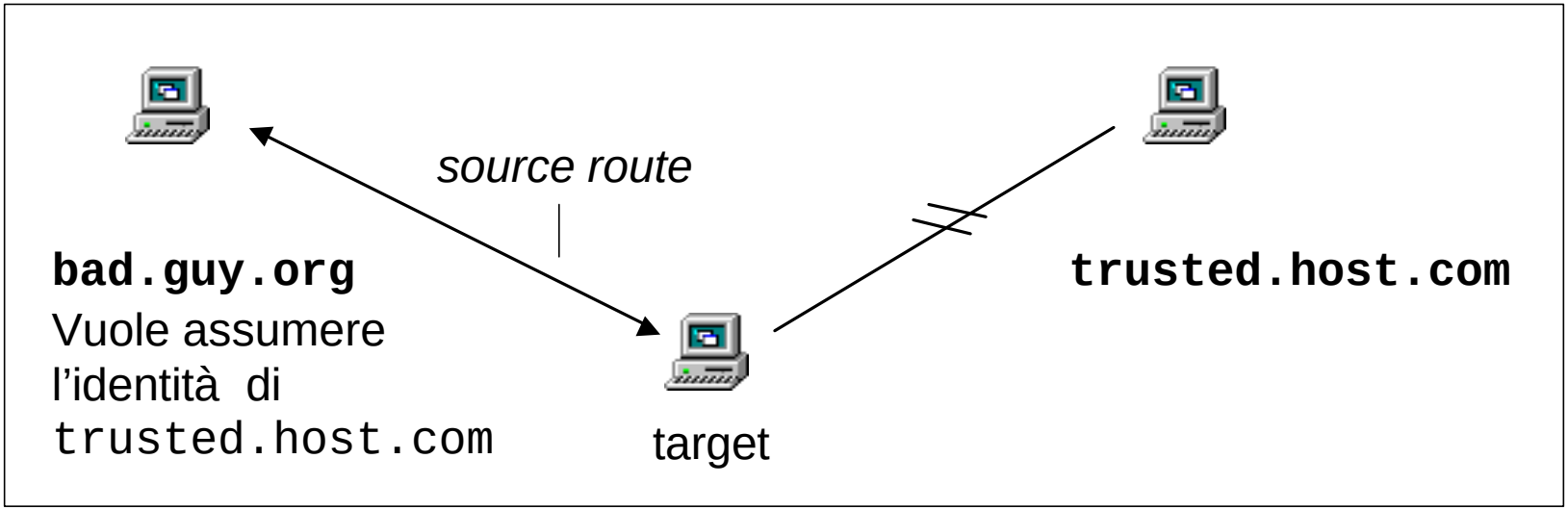
Anti-Spoofing via Unicast RPF

Il controllo di ammissibilità per ogni pacchetto è fatto implicitamente in ingresso all'interfaccia. Se all' IP sorgente non è associata una route nella "*CEF table*" che punta a ritroso (*reverse path*) sulla stessa interfaccia su cui il pacchetto è arrivato esso viene scartato



Source-Routing

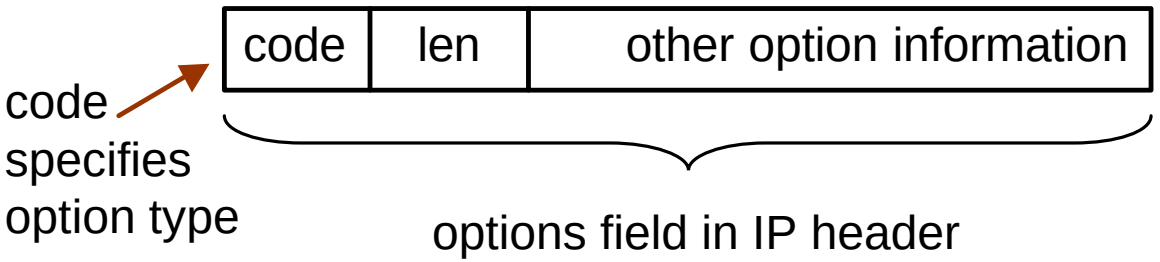
Il meccanismo di **source routing** permette di specificare la strada che i pacchetti dovranno percorrere per giungere a destinazione
Tutta una serie di attacchi tendono ad influenzare i meccanismi di instradamento al fine di effettuare modifiche illecite alla topologia e all'uso delle risorse della rete oppure di perturbarne il funzionamento al solo scopo di denial of service



***Attenzione**:* L'host target utilizzerà le informazioni di source-routing fornite illecitamente da **`bad.guy.org`** per raggiungere l'host **`trusted.host.com`** .
Tutto il traffico per **`trusted.host.com`** viene instradato a **`bad.guy.org`**

Source-Routing – IP Options

- **loose source routing (`code 0x83`)**
  - Specifica una lista di indirizzi IP che devono essere attraversati dal pacchetto
- **strict source routing (`code 0x89`)**
  - Solo gli indirizzi IP nella lista specificata possono essere attraversati dal pacchetto



Source-Routing

Filtri `tcpdump`

Controlla se le IP options sono attive:

```
(ip[0:1] & 0x0f > 5)
```

Discrimina le options loose e strict source routing:

```
and ((ip[20:1] = 0x83) or (ip[20:1] = 0x89))
```

Note: Potrebbe risultare sufficiente anche il solo controllo sulla presenza di IP options attive.

Configurazione Cisco

```
no ip source-route
```

Forza il router a scartare tutti i pacchetti ricevuti con opzioni di source routing.

Routing dinamico - redirects e proxy arp

- Il meccanismo dell'ICMP redirect, in genere usato per informare le stazioni di una rete locale circa l'uso preferenziale di un router per raggiungere determinate destinazioni può essere utilizzato per corrompere opportunamente dall'esterno le tavole di routing degli hosts di una rete. E' opportuno quindi filtrare i redirects in ingresso e inibire la funzionalità a livello di interfaccia

```
access-list 110 deny icmp any any redirect
```

- la funzionalità di proxy arp, attiva di default per permettere al router di rispondere per conto di altri hosts presenti sulla rete connessa può essere utilizzata allo scopo di perturbare l'integrità dell'instradamente. Pertanto è opportuno prevederne la disabilitazione a livello di interfacce esterne.

```
interface Serial0/1
    no ip proxy-arp
```

Protezione del routing dinamico

- Ove sia previsto lo scambio di informazioni di instradamento attraverso protocolli di routing dinamico, è sempre opportuno, a scopo di garantire l'integrità dei processi di routing da alterazioni dolose, prevedere l'uso di meccanismi di autenticazione dei partecipanti al colloquio.
- I protocolli di routing che attualmente offrono il meccanismo della *neighbor authentication* sono:

BGP *	**EIGRP ***
DRP SA	**OSPF ***
IS-IS	**RIPv2 ***

*** = Autenticazione non in chiaro (MD5)**

Protezione del routing dinamico

- Per attivare una sessione BGP con autenticazione MD5

```
router bgp 65282
neighbor 193.206.130.5 remote-as 137
neighbor 193.206.130.5 password mysecret
```

- Analogamente nel caso di OSPF

```
interface Ethernet 0
    ip address 192.133.28.254 255.255.255.0
    ip ospf authentication-key mysecret

router ospf  26
    network 0.0.0.0 255.255.255.255 area 0
    area 0 authentication
```

Protezione del routing dinamico

- E’ inoltre opportuno non accettare, ne’ inviare nei routing updates classi riservate RFC 1918. È pertanto consigliabile operare meccanismi di filtraggio degli annunci (in e out) a livello di liste di distribuzione

```
router bgp 65282
neighbor 193.206.130.5 remote-as 137
neighbor 193.206.130.5 distribute-list 107 in
neighbor 193.206.130.5 distribute-list 108 out

access-list 108 deny ip host 0.0.0.0 any
access-list 108 deny ip 10.0.0.0 0.255.255.255 255.0.0.0
  0.255.255.255
access-list 108 deny ip 172.16.0.0 0.15.255.255 255.240.0.0
  0.15.255.255
access-list 108 deny ip 192.168.0.0 0.0.255.255 255.255.0.0
  0.0.255.255
access-list 108 permit ip any any
```

Protezione NTP

- E’ possibile garantire l’integrità delle informazioni di sincronizzazione temporale ottenute attraverso il protocollo NTP tramite autenticazione basata su MD5

```
clock timezone GMT +1
clock summer-time zone recurring
ntp authenticate
ntp authentication-key 1 md5 ntpk3y
ntp trusted-key 1
ntp access-group peer 20
ntp server ntp_server1_ip key 1 prefer
ntp server ntp_server2_ip key 1
access-list 20 permit host ntp_server1_ip
access-list 20 permit host ntp_server2_ip
```

NETWORK SCANNING

Tecniche di Network Mapping
Port Scanning

CARATTERIZZAZIONE
INDIVIDUAZIONE
DIFESE E CONTROMISURE

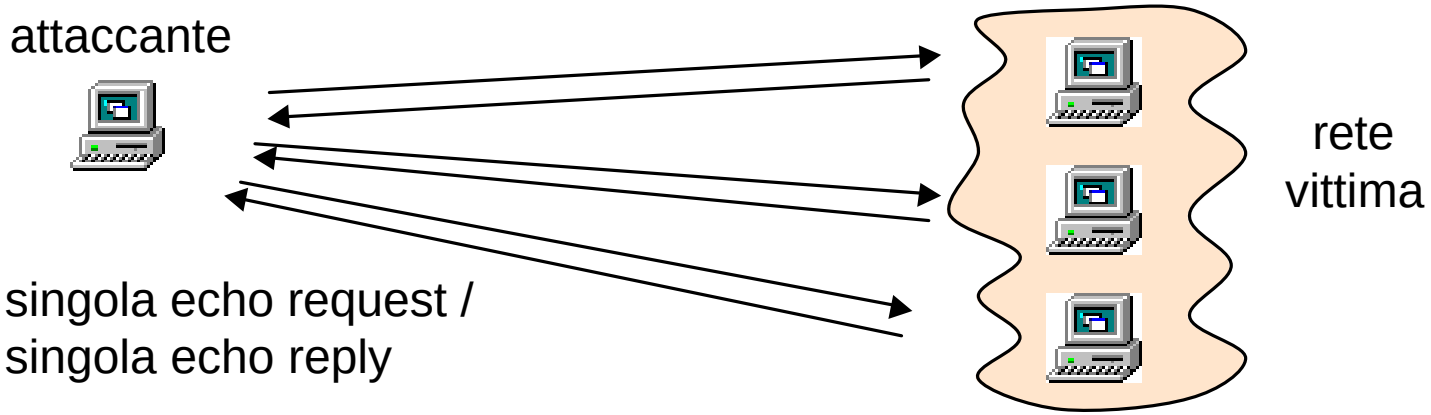
Network mapping

- Il primo passo nell'individuazione dei sistemi vulnerabili presenti su di una rete è quello di individuare tutti gli hosts attivi
- Tecniche più sofisticate consentono anche di determinare dall'esterno la struttura della rete vittima per individuarne i maggiori punti di vulnerabilità
- Ottenute teli informazioni è possibile procedere con scansioni di dettaglio per individuare i servizi attivi sulle macchine presenti sulla rete ed esplorare le loro eventuali vulnerabilità

UDP Network Mapping

Tramite richieste UDP echo (port 7)

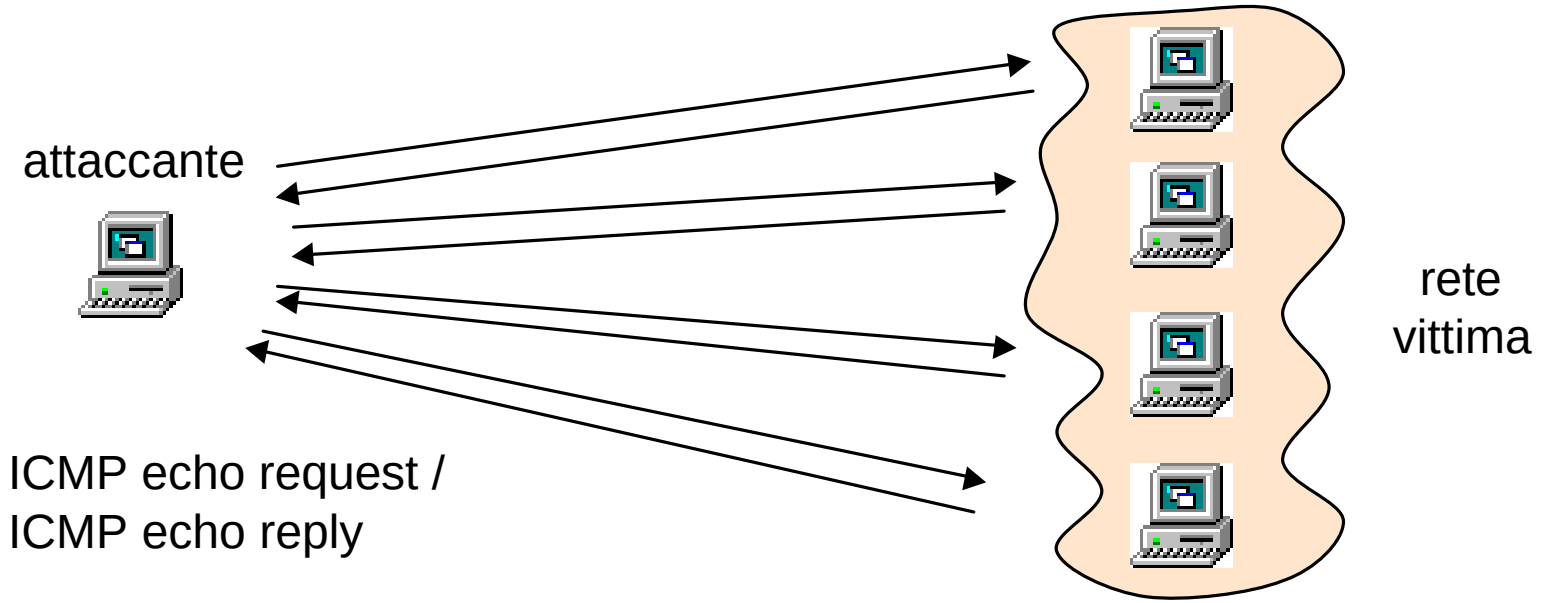
```
02:08:48.088681 slowpoke.mappem.com.3066 > 192.168.134.117.echo: udp 6
02:15:04.539055 slowpoke.mappem.com.3066 > 172.31.73.1.echo: udp 6
02:15:13.155988 slowpoke.mappem.com.3066 > 172.31.16.152.echo: udp 6
02:22:38.573703 slowpoke.mappem.com.3066 > 192.168.91.18.echo: udp 6
02:27:07.867063 slowpoke.mappem.com.3066 > 172.31.2.176.echo: udp 6
02:30:38.220795 slowpoke.mappem.com.3066 > 192.168.5.103.echo: udp 6
02:49:31.024008 slowpoke.mappem.com.3066 > 172.31.152.254.echo: udp 6
02:49:55.547694 slowpoke.mappem.com.3066 > 192.168.219.32.echo: udp 6
03:00:19.447808 slowpoke.mappem.com.3066 > 172.31.158.86.echo: udp 6
03:05:29.484029 slowpoke.mappem.com.3066 > 192.168.191.108.echo: udp 6
03:23:24.348176 slowpoke.mappem.com.3066 > 192.168.226.120.echo: udp 6
03:24:15.755411 slowpoke.mappem.com.3066 > 172.31.173.5.echo: udp 6 ...
```



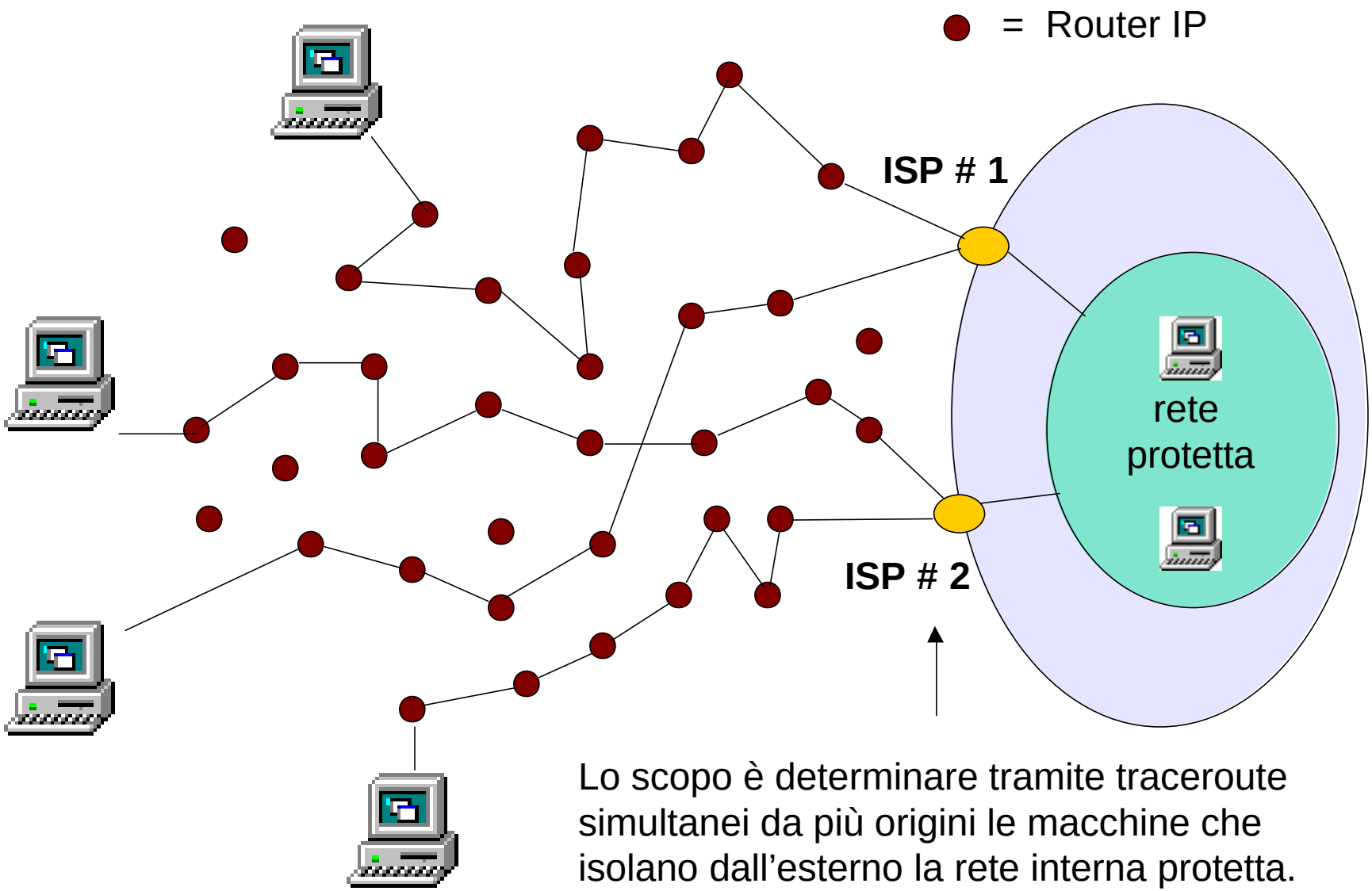
ICMP Network Mapping

Invio ICMP-echo a una macchina alla volta con indirizzo sorgente non spoofato

```
01:00:38.861865 pinger.mappem.com > 192.168.6.1: icmp: echo request
01:00:51.903375 pinger.mappem.com > 192.168.6.2: icmp: echo request
01:01:04.925395 pinger.mappem.com > 192.168.6.3: icmp: echo request
01:01:18.014343 pinger.mappem.com > 192.168.6.4: icmp: echo request
01:01:31.035095 pinger.mappem.com > 192.168.6.5: icmp: echo request
01:01:44.078728 pinger.mappem.com > 192.168.6.6: icmp: echo request
01:01:57.098411 pinger.mappem.com > 192.168.6.7: icmp: echo request ...
```



ICMP traceroute Network Mapping



Lo scopo è determinare tramite traceroute simultanei da più origini le macchine che isolano dall'esterno la rete interna protetta.

ICMP traceroute Network Mapping

Serie di traceroute simultanei da più provenienze

```
10:32:24.722 north.mappem.com.38758 > ns.target.net.33476: udp 12
10:32:24.756 north.mappem.com.38758 > ns.target.net.33477: udp 12
10:32:24.801 north.mappem.com.38758 > ns.target.net.33478: udp 12
10:32:24.833 north.mappem.com.38758 > ns.target.net.33479: udp 12
10:32:24.944 north.mappem.com.38758 > ns.target.net.33481: udp 12

10:32:26.541 south.mappem.com.48412 > ns.target.net.33510: udp 12
10:32:26.745 south.mappem.com.48412 > ns.target.net.33512: udp 12
10:32:26.837 south.mappem.com.48412 > ns.target.net.33513: udp 12
10:32:26.930 south.mappem.com.48412 > ns.target.net.33514: udp 12
10:32:27.033 south.mappem.com.48412 > ns.target.net.33515: udp 12

10:32:26.425 east.mappem.com.58853 > ns.target.net.33490: udp 12
10:32:26.541 east.mappem.com.58853 > ns.target.net.33491: udp 12
10:32:26.744 east.mappem.com.58853 > ns.target.net.33493: udp 12
10:32:26.836 east.mappem.com.58853 > ns.target.net.33494: udp 12
10:32:26.930 east.mappem.com.58853 > ns.target.net.33495: udp 12
10:32:27.033 east.mappem.com.58853 > ns.target.net.33496: udp 12
```

Traceroute Network Mapping: Firewalking

Attraverso una tecnica più sofisticata, detta “firewalking”, basata sull’uso dei TTL e dei messaggi di errore ICMP, è possibile acquisire informazioni di dettaglio circa la struttura di una rete interna protetta da un firewall ed in particolare circa le politiche di packet filtering operate dal firewall stesso

```
# firewalk –n –P1-8 –pTCP 10.0.0.5 10.0.0.20
Firewalking through 10.0.0.5 (towards 10.0.0.20) with a maximum of 25 hops.
Ramping up hopcounts to binding hosts…
probe:  1  TTL:  1  port 33434: <response from> [10.0.0.1]
probe:  2  TTL:  1  port 33434: <response from> [10.0.0.2]
probe:  3  TTL:  1  port 33434: <response from> [10.0.0.3]
probe:  4  TTL:  1  port 33434: <response from> [10.0.0.4]
probe:  5  TTL:  1  port 33434: Bound scan: 5 hops <Gateway at 5 hops> [10.0.0.5]
port    1: open
port    2: open
… …
```

Ad esempio, seguito dello scarto di un pacchetto dovuto alla violazione di una regola prevista in un’ACL, il router invia a ritroso il messaggio *ICMP administratively prohibited* (tipo 3 code 13) che può essere utilizzato da chi attacca per ricavare informazioni circa la politica di filtraggio.

ICMP traceroute Network Mapping

Filtri `tcpdump`:

```
udp and (udp[2:2] >= 33000) and (udp[2:2] <= 34999)
```

Contromisure (Cisco ACL):

Blocco in ingresso dei traceroute

```
access-list 110 deny udp any 172.16.0.0 0.0.255.255 gt 30000
access-list 111 deny icmp 172.16.0.0 0.0.255.255 any time-exceeded
access-list 111 deny icmp 172.16.0.0 0.0.255.255 any unreachable
```

Consenti i traceroute in uscita

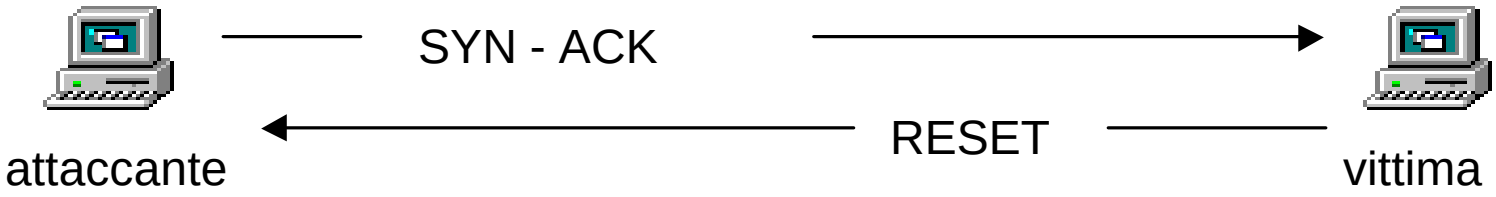
```
access-list 111 permit udp 172.16.0.0 0.0.255.255 any gt 30000
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 time-exceeded
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 unreachable
```

Inibisci a livello di router la generazione dei "ICMP unreachable"

```
interface serial 6/0
       no ip unreachables
```

TCP Stealth Network mapping

Per individuare la presenza di hosts attivi sulla rete è possibile utilizzare, una tecnica basata sul protocollo TCP molto più subdola e **meno evidente** rispetto a quelle che ricorrono a meccanismi (UDP, ICMP o TCP) di echo request/reply:



Ogno qual volta un host riceve un pacchetto SYN-ACK su una qualsiasi porta, lo stesso risponde con un RST, indipendentemente che la porta sia aperta o meno

```
11:37:50.065 attacker > 172.21.165.1: icmp: echo request
11:37:50.065 attacker.62072 > 172.21.165.1.80: . ack 0 win 3072
11:37:50.075 172.21.165.1 > attacker: icmp: echo reply
11:37:50.075 172.21.165.1.80 > attacker.62072: R 0:0(0) win 0
```

Contromisure: La regola di filtraggio *established* realizza un bocco efficace

```
access-list 110 permit any 172.1.2.0 0.0.0.255 established
```

Filtro tcpdump:

```
tcp and (tcp[13] & 0x10 != 0) and
        (tcp[13] & 0x04 = 0) and (tcp[8:4] = 0)
```

Portscan

- L’originatore della scansione effettua da remoto un test dello stato delle porte TCP o UDP su un host remoto per determinare le porte attive (aperte), che ammettono connessioni in ingresso, e le porte non utilizzate (chiuse), che non ammettono connessioni

- Una porta aperta individua la presenza di un servizio di rete attivo offerto dall’host target

- A partire dall’elenco dei servizi offerti l’attaccante può tentare di individuare eventuali vulnerabilità conosciute ed effettuare exploits sulle stesse

UDP Portscan

Le porte chiuse rispondono con un errore ICMP “port unreachable”

Porta chiusa

```
11:40:36.445995 attacker.org.53160 > target.com.516: udp 0
11:40:36.455995 target.com > attacker.org:
                  icmp: 172.21.165.150 udp port 516 unreachable
```

Le porte aperte non rispondono in alcun modo

Porta aperta

```
11:40:36.855995 attacker.org.53160 > target.com.514: udp 0
11:40:37.005995 attacker.org.53161 > target.com.514: udp 0
11:40:37.165995 attacker.org.53160 > target.com.514: udp 0
11:40:37.255995 attacker.org.53161 > target.com.514: udp 0
```

- Si assume che tutte le porte che non rispondono al probe siano aperte
- Va tenuto presente che UDP è un protocollo “unreliable” (come ICMP)
- Questo rende l’UDP portscan inaffidabile e difficoltoso quando la destinazione risulta essere piuttosto lontana in termini di hop count

Connect TCP scan

Porta aperta (ammette conessioni)

```
scanner.8831 > target.514: S 3209086149:3209086149(0)
target.514 > scanner.8831: S 1346112000:1346112000(0) ack 3209086150
scanner.8831 > target.514: . ack 1346112001
scanner.8831 > target.514: F 3209086150:3209086150(0) ack 1346112001
target.514 > scanner.8831: . ack 3209086151
scanner.514 > target.8831: F 1346112001:1346112001(0) ack 3209086151
target.8831 > scanner.514: . ack 1346112002
```

Porta chiusa (non ammette connesioni)

```
scanner.12441 > target.516: S 1573861375:1573861375(0)
target.516 > scanner.12441: R 0:0(0) ack 1573861376
```

- Il three-way handshake viene completato all'apertura di una porta che accetta la connessione che poi viene chiusa regolarmente con una "close request".
- Se una porta è chiusa, la vittima risponde alla richiesta di connessione con un RESET
- In genere questi tentativi di connessione **sono oggetto di logging**

SYN TCP scan

Porta aperta (accetta connessioni)

```
scanner.52894 > target.514: S 3900690976:3900690976(0)
target.514 > scanner.52894: S 1379776000:1379776000(0) ack 3900690977
scanner.52894 > target.514: R 3900690977:3900690977(0)
```

Porta chiusa (non accetta connessioni)

```
scanner.52894 > target.516: S 3900690976:3900690976(0)
target.516 > scanner.52894: R 0:0(0) ack 3900690977
```

- In questa tecnica il three-way handshake non viene completato al set-up
- Lo scanner invia un pacchetto di SYN costruito ad hoc e attende la risposta
- Se la vittima risponde con un SYN-ACK, ammettendo la connessione sulla porta scansita, il SO dell'attaccante cessa immedatamente la connessione, non avendo effettuato da parte sua una regolare apertura della stessa
- Generalmente tali tentativi **non sono oggetto di logging** non avendo completato l'handshaking iniziale

SYN e Connect TCP Scan

Individuazione connect scan

Le porte vengono scansite in sequeza, sequence number e src port cambiano

```
11:17:59 scanner.29699 > target.264: S 884860893:884860893(0)
11:17:59 scanner.29700 > target.265: S 2647868987:2647868987(0)
11:17:59 scanner.29720 > target.266: S 3719918849:3719918849(0)
```

Individuazione SYN scan

La porta sorgente e i sequence number non cambiano: i pacchetti sono costruiti

```
11:22:14.38 scanner.52894 > target.386: S 3900690976:3900690976(0)
11:22:14.38 scanner.52894 > target.338: S 3900690976:3900690976(0)
11:22:14.38 scanner.52894 > target.369: S 3900690976:3900690976(0)
```

Tecniche di difesa

La regola di filtraggio *established* blocca efficacemente entrambe le tecniche

```
access-list 110 permit any 172.21.0.0 0.0.255.255 established
```

Filtro `tcpdump`

Traccia tutte le connessioni SYN in ingresso a porte su cui non ci si aspetta traffico

```
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x10 = 0) and
(not dst port 53) and (not dst port 80)
and (not dst port 25) and (not dst port 21)
```

Non-SYN-ACK-RST TCP scan

Le tre tecniche fondamentali:
FIN scan , Xmastree scan e Null scansi comportano identicamente
(l'esempio si riferisce alla FIN-scan)

Porta aperta (ammette connessioni)

```
11:24:52.545 scanner.org.57298 > target.com.514: F 0:0(0)
11:24:52.655 scanner.org.57299 > target.com.514: F 0:0(0)
11:24:53.445 scanner.org.57298 > target.com.514: F 0:0(0)
11:24:53.535 scanner.org.57299 > target.com.514: F 0:0(0)
```

Porta chiusa (non ammette connessioni)

```
11:24:52.495 scanner.org.57298 > target.com.516: F 0:0(0)
11:24:52.495 target.com.516 > scanner.org.57298: R 0:0(0) ack 0
```

• Per la RFC 793: "TCP Functional Specification" un pacchetto non di RST inviato su una porta chiusa deve causare l'invio di un RST a ritroso

• I sistemi MS Windows, Cisco IOS, BSDI, HP/UX, MVS, IRIX, che non rispettano a pieno la RFC 793 non sono soggetti a questo tipo di scansione

Non-SYN-ACK-RST TCP scan

Scan con invio di FIN (FIN scan)

```
11:24:51.975 scanner.org.57298 > target.com.699: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.410: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.876: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.363: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.215: F 0:0(0)
```

Scan con invio di FIN e flags PUSH, URGENT (Xmastree scan)

```
11:30:48.065 scanner.org.38674 > target.com.895: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.56: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.299: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.888: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.267: FP 0:0(0) urg 0
```

Scan con invio di pacchetti Null (senza flags)

```
11:33:36.225 scanner.org.63816 > target.com.821: .
11:33:36.225 scanner.org.63816 > target.com.405: .
11:33:36.225 scanner.org.63816 > target.com.391: .
11:33:36.225 scanner.org.63816 > target.com.59: .
11:33:36.225 scanner.org.63816 > target.com.91: .
```

Non-SYN-ACK-RST TCP scan

SYN flag:	`tcp[13] & 0x02 != 0`
ACK flag:	`tcp[13] & 0x10 != 0`
RST flag:	`tcp[13] & 0x04 != 0`
FIN flag:	`tcp[13] & 0x01 != 0`
no flags:	`tcp[13] & 0x3f = 0`

Nessun flag settato
```
tcp and (tcp[13] & 0x3f = 0)
```

Tecniche di difesa (Cisco ACL)
La regola *established* blocca le scansioni

FIN flag settato e ACK flag non settato
```
tcp and (tcp[13] & 0x01 != 0) and (tcp[13] & 0x10 = 0)
```

SYN flag e FIN flag simultaneamente settati
```
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x01 != 0)
```

RST flag e FIN flag simultaneamente settati
```
tcp and (tcp[13] & 0x04 != 0) and (tcp[13] & 0x01 != 0)
```

SYN flag e RST flag simultaneamente settati
```
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x04 != 0)
```

TCP Decoy scan

La reale provenienza della scansione è mascherata attraverso l'invio di un'enorme numero di altri pacchetti di scansione da indirizzi spoofati

```
06:43:55 10.2.2.2.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.280: S 1496167267:1496167267(0)
```

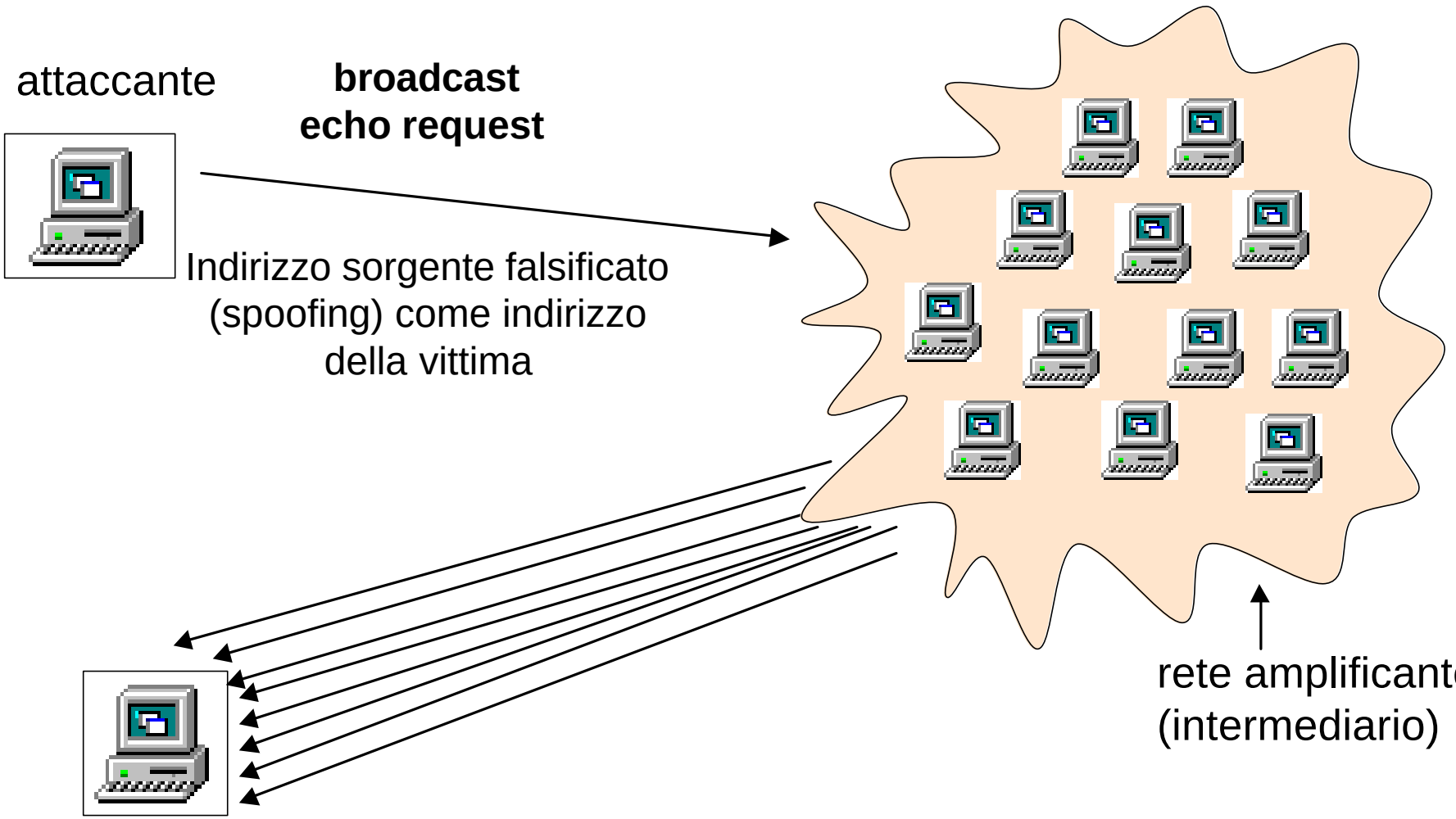
Chi è il reale autore della scansione?

DENIAL OF SERVICE

Tassonomia e analisi delle principali tecniche di attacco

CARATTERIZZAZIONE
INDIVIDUAZIONE
DIFESE E CONTROMISURE

Smurfing - Fraggle



echo replies ricevuti dalla vittima in ragione delle risposte ai pacchetti di broadcast inviati dall'indirizzo della vittima a tutti i componenti della rete dell'intermediario

Smurfing - Fraggle

L’aggressore invia flussi di traffico verso indirizzi di broadcast attribuendosi come indirizzo sorgente quello della vittima. Se i router sulle reti di destinazione propagano i broadcast IP al livello 2 tutti gli host su tali reti risponderanno all’indirizzo falsificato con un echo-reply generando a ritroso un flusso di traffico pari a quello entrante moltiplicato per il loro numero

!blocca il traffico esplicitamente destinato a indirizzi di broadcast

```
access-list 110 deny ip any 0.0.0.255 255.255.255.0
access-list 110 deny ip any 0.0.0.0 255.255.255.0
access-list 110 permit ip any any
```

! Su tutte le interfacce broadcast-capable disabilita la propagazione dei directed-broadcast a livello 2 - tale opzione e’ il default a partire da **IOS 12.0**

```
interface Ethernet0/0
       no ip directed-broadcast
```

Smurfing - Fraggle

Espressioni di filtraggio **tcpdump**:

Broadcast **network.255**: **ip and ip[19] = 0xff**
Broadcast **network.0** : **ip and ip[19] = 0x00**

```
00:00:05.327 spoofed.target.com > 192.168.15.255: icmp: echo request
00:00:05.342 spoofed.target.com > 192.168.1.255:  icmp: echo request
00:00:14.154 spoofed.target.com > 192.168.15.255: icmp: echo request
00:00:14.171 spoofed.target.com > 192.168.1.255:  icmp: echo request

05:20:48.261 spoofed.target.com > 192.168.0.0:     icmp: echo request
05:20:48.263 spoofed.target.com > 255.255.255.255: icmp: echo request
05:21:35.792 spoofed.target.com > 192.168.0.0:     icmp: echo request
05:21:35.819 spoofed.target.com > 255.255.255.255: icmp: echo request
```

Smurfing usa per gli attacchi l'ICMP echo request/reply
Fraggle usa UDP echo request/reply

Landing

Il “land” o TCP loopback DoS è basato sull’invio di TCP SYN con indirizzo e porta sorgente falsificati e impostati identici a indirizzo e porta di destinazione. Questo può causare per release IOS meno recenti il blocco totale del router

L’applicazione di una regola/ACL di filtraggio anti-spoofing previene Completamente a possibilità di tali attacchi dall’esterno:

```
access-list 110 deny ip 192.4.1.0 0.255.255.255 any
```

Per un’attacco in corso dall’interno l’unica possibilità è il filtraggio diretto dei Pacchetti che caratterizzano l’attacco

```
access-list 110 deny ip host 192.4.1.1 host 192.4.1.1
access-list 110 permit ip any any
```

Landing

Caratterizzazione:

- IP sorgente = IP destinazione (sorgente spoofed)
- port sorgente = port destinazione
- Pacchetti TCP packet con il SYN flag settato
- Port aperta sull'host target

Filtri `tcpdump`

```
ip[12:4] = ip[16:4]
```

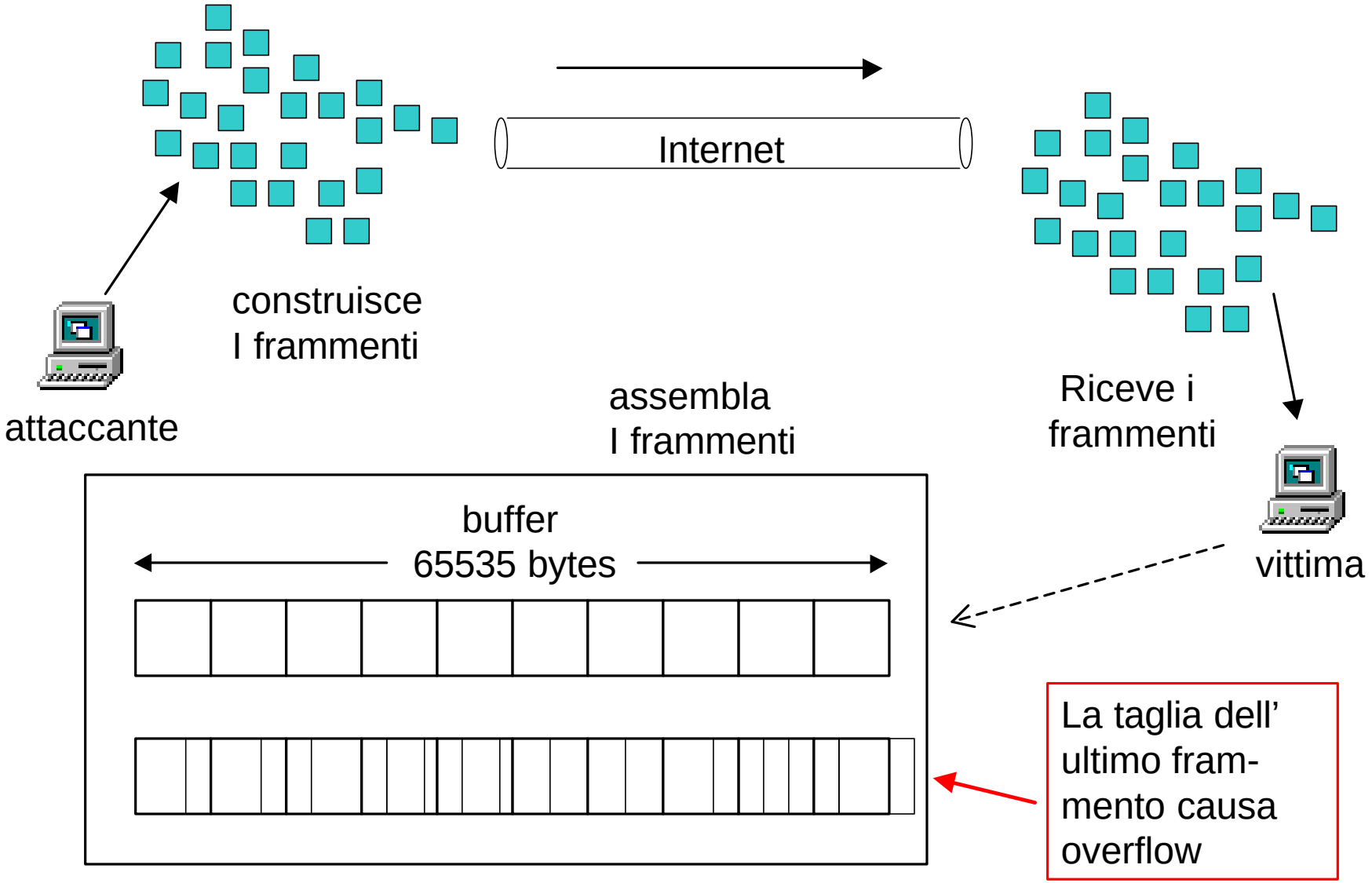
Rileva i pacchetti con indirizzo sorgente e di destinazione uguali

```
ip[12:2] = ip[16:2]
```

Rileva i pacchetti con indirizzi di network sorgente e destinazione uguali

```
10:56:32.395383 gamma1.victim.net.139 > gamma1.victim.net.139: S
10:56:35.145383 gamma1.victim.net.139 > gamma1.victim.net.139: S
10:56:36.265383 gamma1.victim.net.139 > gamma1.victim.net.139: S
```

Ping o’ Death

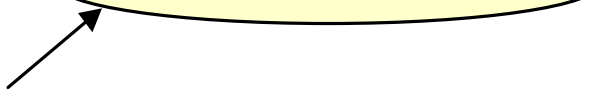

Internet
construisce
I frammenti
attaccante
assembla
I frammenti
Riceve i
frammenti
vittima
buffer
65535 bytes
La taglia dell’
ultimo fram-
mento causa
overflow

Ping o’ Death

Filtri `tcpdump`:

`icmp and (ip[6:1] & 0x20 !=0)and (ip[6:2] & 0x1fff = 0)`

```
12:43:58.431 big.pinger.org > www.mynetwork.net:
                              icmp: echo request (frag 4321:380@0+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@2656+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@760+)
...
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@63080+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@64216+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@65360)
```

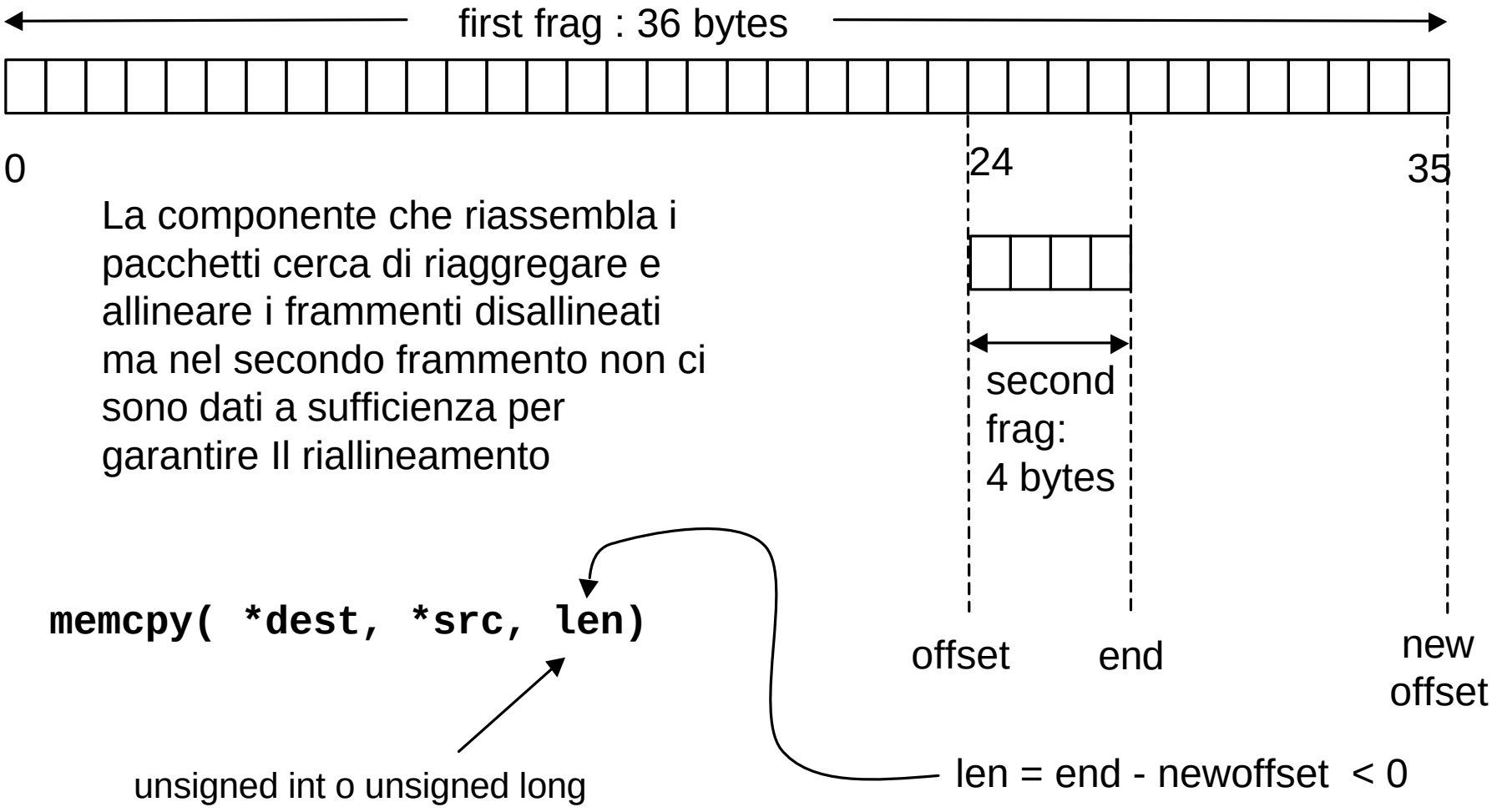


L’aggressore invia un pacchetto ICMP ping più grande della massima taglia consentita per i pacchetti IP: 65535 bytes (380 + 65360 = 65740).

Caratterizzazione dell’attacco:

- pacchetti ICMP con il flag MF settato e il campo fragment-offset a zero
- la dimensione totale dei pacchetti riassemblati supera 65535 bytes

Teardrop


first frag : 36 bytes
0
24
35
La componente che riassembla i pacchetti cerca di riaggregare e allineare i frammenti disallineati ma nel secondo frammento non ci sono dati a sufficienza per garantire Il riallineamento
second frag: 4 bytes
memcpy(*dest, *src, len)
offset
end
new offset
unsigned int o unsigned long
len = end - newoffset < 0

Teardrop

```
10:25:48 attacker.org.45959 > target.net.53: udp 28 (frag 242:36@0+)
10:25:48 attacker.org > target.net: (frag 242:4@24)
```

Caratterizzazione dell'attacco:
2 frammenti di pacchetti UDP:
primo frammento: 0+ frammento con taglia payload = N
secondo frammento: frammento finale con offset < N e taglia payload < (N-offset)

Specific Signature
UDP packet
port open on target host

Result of a successful attack
target machine reboots or halts - depending on the amount of physical memory

`tcpdump` filter:

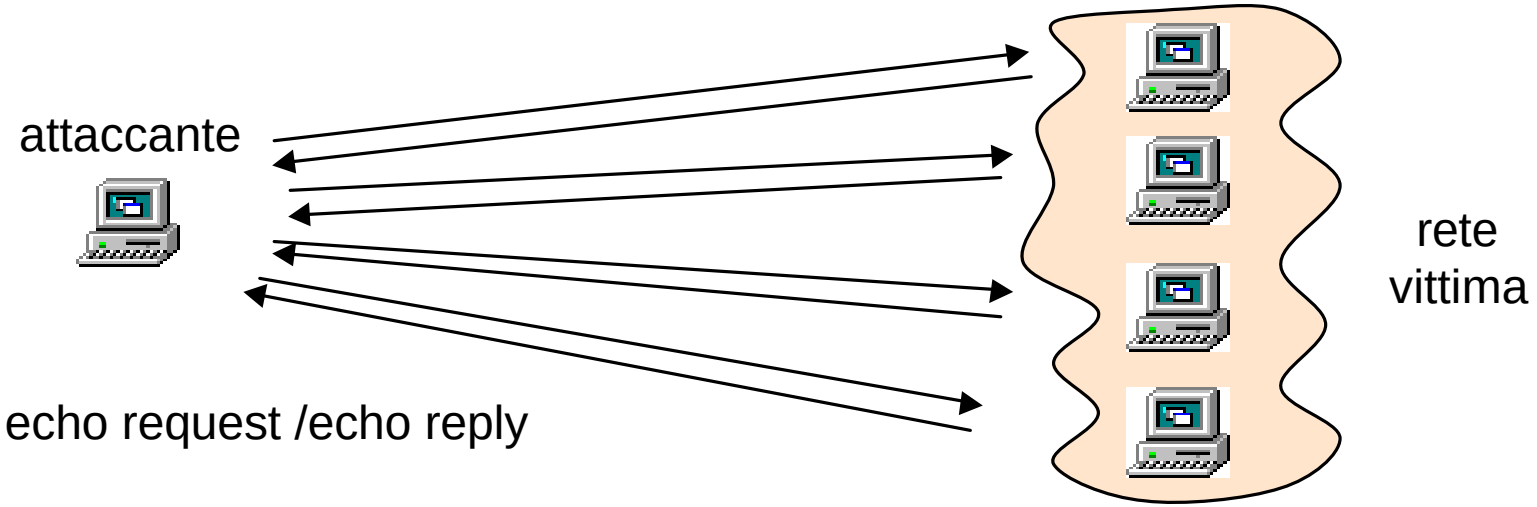
```
udp and (ip[6:1] & 0x20 != 0)
```

Stateful device: does the signature match the general signature given?

ICMP Bombing

Tecnica di flooding che prevede il congestionamento di linee e il sovraccarico elaborativo di routers e hosts attraverso l'invio massivo e indiscriminato di messaggi ICMP (in genere HOST-UNREACHABLE o NETWORK-UNREACHABLE, più raramente anche ECHO request)

```
01:00:38.861865 pinger.mappem.com > 192.168.6.1: icmp: echo request
01:00:38.903375 pinger.mappem.com > 192.168.6.2: icmp: echo request
01:00:39.925395 pinger.mappem.com > 192.168.6.1: icmp: echo request
01:00:39.014343 pinger.mappem.com > 192.168.6.1: icmp: echo request
01:00:39.035095 pinger.mappem.com > 192.168.6.2: icmp: echo request
```



ICMP Bombing – Filtraggio in banda

E’ possibile prevenire o reagire ad attacchi basati sull’ICMP bombing limitando in banda i flussi di traffico offensivi (ICMP) tramite la QoS facility “Committed Access Rate” (CAR) integrata nell’ambito dei meccanismi CEF e “DISTRIBUTED CEF”

Limita il solo traffico ICMP consentito

```
access-list 102 permit icmp any any
```

Applica il filtro in banda (8Kbps) sulla border interface

```
interface Serial3/0/0
rate-limit input access-group 102 256000 8000 8000
conform-action transmit exceed-action drop
```

Si può limitare solo il traffico relativo a ICMP ECHO e UNREACHABLE

```
access-list 102 permit icmp any any echo
access-list 102 permit icmp any any echo-reply
access-list 102 permit icmp any any unreachable
```

ICMP Bombing – Traffic shaping

Il Generic Traffic Shaping (GTS) è un meccanismo di packet filtering di tipo *token-bucket* che permette di imporre a un flusso di traffico IP un throughput massimo inferiore a quello nominale relativo all'interfaccia del router attraverso cui avviene la trasmissione. Tale meccanismo può essere utilizzato per prevenire DoS di flooding predimensionando opportunamente la banda riservata al traffico sospetto

```
access-list 102 permit icmp any any
```

Al traffico ICMP non va garantito più di 1Mb

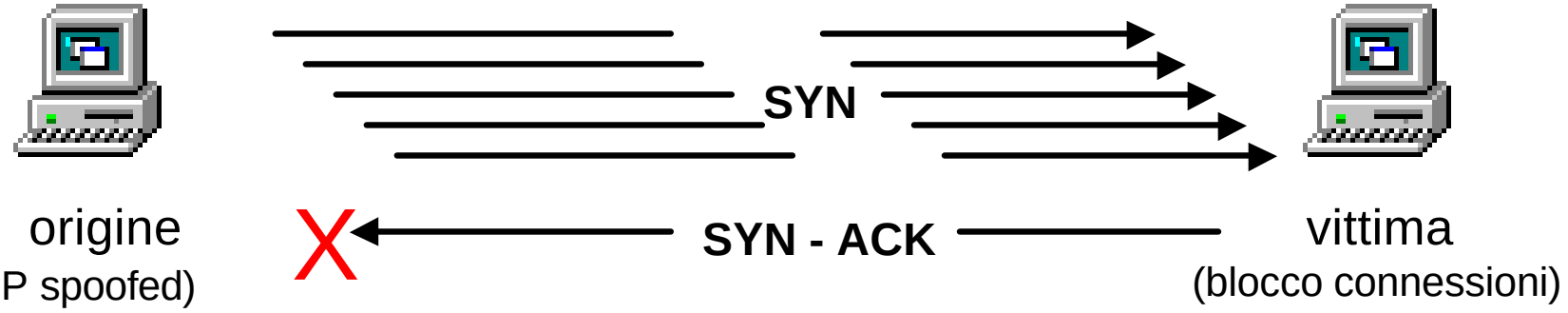
```
interface Serial0 traffic-shape group 101 1000000
125000 125000
```

Anche l'uso del W*eighted Fair Queueing* si rivela generalmente piuttosto efficace per migliorare la tolleranza ai flooding

```
interface Serial 3/0
    ip unnumbered Ethernet 0/0
    fair-queue 64
```

TCP SYN Flooding

Il SYN flooding è una tecnica di DoS caratterizzata dall'apertura di un elevato numero di connessioni da indirizzi diversi, ovviamente falsificati, verso la vittima, curando di evitare l'*ACK* di chiusura del *TCP three way handshake* al fine di saturarne la coda di connessione




```
04:37:19 10.10.10.13.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.14.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.15.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.16.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.17.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.18.41508 > target.23: S 3935335593:3935335593(0)
04:37:19 10.10.10.19.41508 > target.23: S 3935335593:3935335593(0)
```

L'origine dell'attacco viene fatta corrispondere a un indirizzo inesistente in modo che la vittima non riceverà mai a ritroso gli ACK-SYN-ACK generati a fronte dei SYN

TCP SYN Flooding

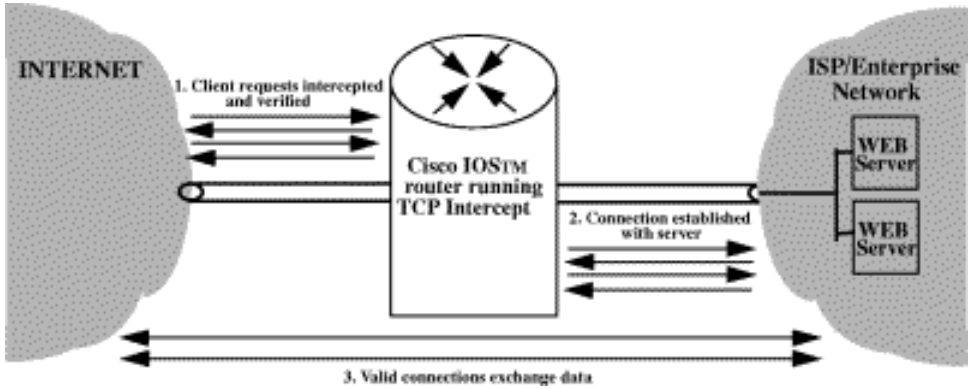
```
TCP
Local Address          Remote Address         State
-------------------- -------------------- -------
*.*                    *.*                  IDLE
*.sunrpc               *.*                  LISTEN
*.ftp                  *.*                  LISTEN
*.telnet               *.*                  LISTEN
*.finger               *.*                  LISTEN
target.telnet    10.10.10.11.41508          SYN_RCVD
target.telnet    10.10.10.12.41508          SYN_RCVD
target.telnet    10.10.10.13.41508          SYN_RCVD
target.telnet    10.10.10.14.41508          SYN_RCVD
target.telnet    10.10.10.10.41508          SYN_RCVD
target.telnet    10.10.10.15.41508          SYN_RCVD
target.telnet    10.10.10.16.41508          SYN_RCVD
target.telnet    10.10.10.17.41508          SYN_RCVD
target.telnet    10.10.10.18.41508          SYN_RCVD
target.telnet    10.10.10.20.41508          SYN_RCVD
*.*                    *.*                  IDLE
```

Output di **`netstat -a`** sull'host vittima

Una volta che la specifica coda al livello di TCP stack è completamente saturata dalle connessioni in fase di setup qualsiasi apertura di un TCP socket verso la vittima diventa impossibile

TCP SYN Flooding – TCP Intercept

Disponibile in IOS 11.2(4)F, 11.3 e successive, previene i SYN-flood intercettando e validando, in modalità “proxy” le richieste di connessione TCP verso gli hosts, definiti tramite ACL, aprendo una semiconnessione col client successivamente estesa al server in caso di successo



Abilita il TCP intercept per le reti definite via ACL

```
ip tcp intercept list 101

access-list 101 permit tcp any 192.133.28.0 0.0.0.255
```

TCP SYN Flooding – Filtraggio in banda

E’ possibile reagire attivamente durante un attacco di tipo “SYN flooding” per ridurne drasticamente l’impatto, limitando in banda il flusso di traffico offensivo tramite la QoS facility “Committed Access Rate” (CAR) integrata nell’ambito dei meccanismi CEF e “DISTRIBUTED CEF”

Non influenzare le sessioni TCP già completamente stabilite

```
access-list 103 deny tcp any host 10.0.0.1 established
```

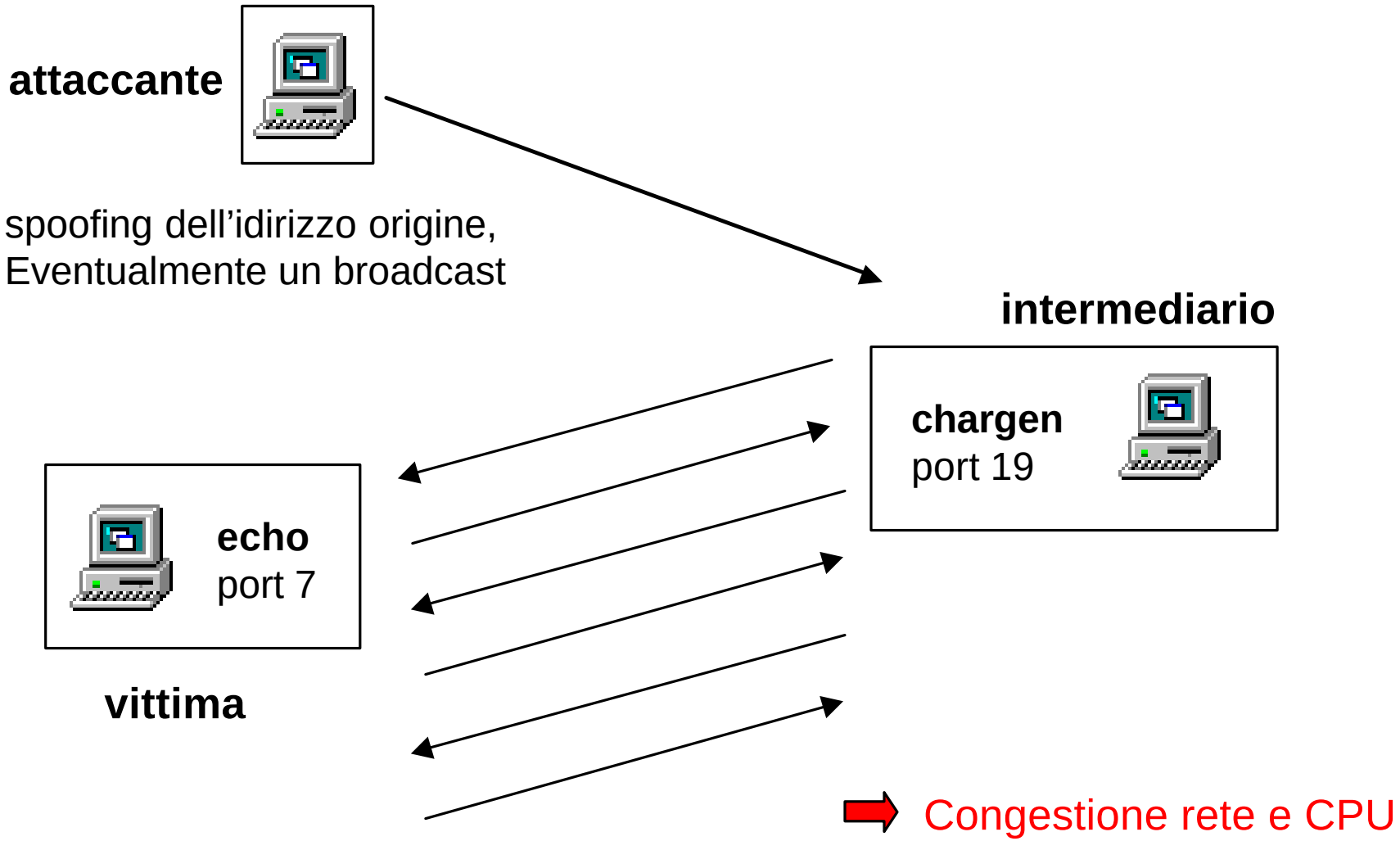
Limita in banda tutto il restante traffico (le sessioni in SYN)

```
access-list 103 permit tcp any host 10.0.0.1
```

Applica il filtro in banda (8Kbps) sulla border interface

```
interface Serial3/0/0
   rate-limit input access-group 103 8000 8000 8000
   conform-action transmit exceed-action drop
```

Diagnostic Port DoS



Diagnostic Port DoS

nome	port number	descrizione del servizio
echo	7/udp 7/tcp	il server restituisce in echo a ritroso quanto inviato dal client
discard	9/udp 9/tcp	il server scarta in modo silente quanto inviato dal client
daytime	13/udp 13/tcp	il server restituisce ora e data in formato leggibile
chargen	19/udp 19/tcp	il server risponde con un datagramma contenente una stringa di caratteri ascii il server invia uno stream continuo di caratteri finchè la connessione non viene terminata dal client
time	37/udp 37/tcp	il server restituisce ora e data in formato binario a 32 bit

Diagnostic Port DoS

L’invio di elevate quantità di traffico TCP o UDP sulle porte di diagnostica del router (*echo, discard, chargen, daytime*) può avere un notevole impatto sia sulla rete che sul carico elaborativo del router stesso, fino a degradarne le prestazioni o causarne, in condizioni estreme il blocco

! E’ consigliabile disabilitare a livello di IOS tutti i servizi “diagnostic port”

```
no service udp-small-servers
no service tcp-small-servers
```

udp-small-servers: echo, discard, chargen
tcp-small-servers: echo, chargen, discard, daytime

Servizi abilitati per default sulle release IOS < **12.0**

! Per proteggere anche gli host interni va bloccato tutto il traffico verso i
! servizi “diagnostic port”

```
access-list 110 deny udp any 172.16.0.0 0.0.255.255 eq 7
access-list 110 deny udp any 172.16.0.0 0.0.255.255 eq 13
access-list 110 deny udp any 172.16.0.0 0.0.255.255 eq 19
access-list 110 deny udp any 172.16.0.0 0.0.255.255 eq 37
```

Diagnostic Port DoS

Espressione di filtraggio per `tcpdump`:

```
udp and ( ((port 7) and (port 13)) or ((port 7) and
(port 19)) or ((port 7) and (port 37)) or ((port 13) and
(port 19)) or((port 13) and (port 37)) or ((port 19) and
(port 37)) or ((src port 7) and (dst port 7)) or
((src port 13) and (dst port 13)) or((src port 19) and
(dst port 19)) or ((src port 37) and (dst port 37)) )
```

Un singolo pacchetto avvia l'oscillazione creando il loop infinito

```
08:08:16.155354 spoofed.target.net.echo > 172.31.203.17.chargen: udp
```

Per ottenere un effetto amplificato di ricorre a diversi stream di pacchetti

```
08:08:16.155354 spoofed.target.net.echo > 172.31.203.17.chargen: udp
08:21:48.891451 spoofed.target.net.echo > 192.168.14.50.chargen: udp
08:25:12.968929 spoofed.target.net.echo > 192.168.102.3.chargen: udp
08:42:22.605428 spoofed.target.net.echo > 192.168.18.28.chargen: udp
08:47:21.450708 spoofed.target.net.echo > 172.31.130.93.chargen: udp
08:51:27.491458 spoofed.target.net.echo > 172.31.153.78.chargen: udp
08:53:13.530992 spoofed.target.net.echo > 172.31.46.49.chargen: udp
```

Attacco Cisco IOS Syslog

Il seguente pacchetto può essere causa di crash a livello di alcuni IOS

```
11:40:36.855995 attacker.53160 > router.514: udp 0
```

- L'invio di una serie di datagrammi sulla porta **`syslog`** `del router causa il crash`
- Sono interessate le versioni di IOS identificate come 11.3AA, 11.3DB, e tutte le varianti della 12.0 (12.0S, 12.0T etc.)

Le regole seguenti proteggono la porta in questione risolvendo il problema

Blocca i multicasts diretti alla porta 514

```
access-list 110 deny udp any 224.0.0.0 31.255.255.255 eq 514
```

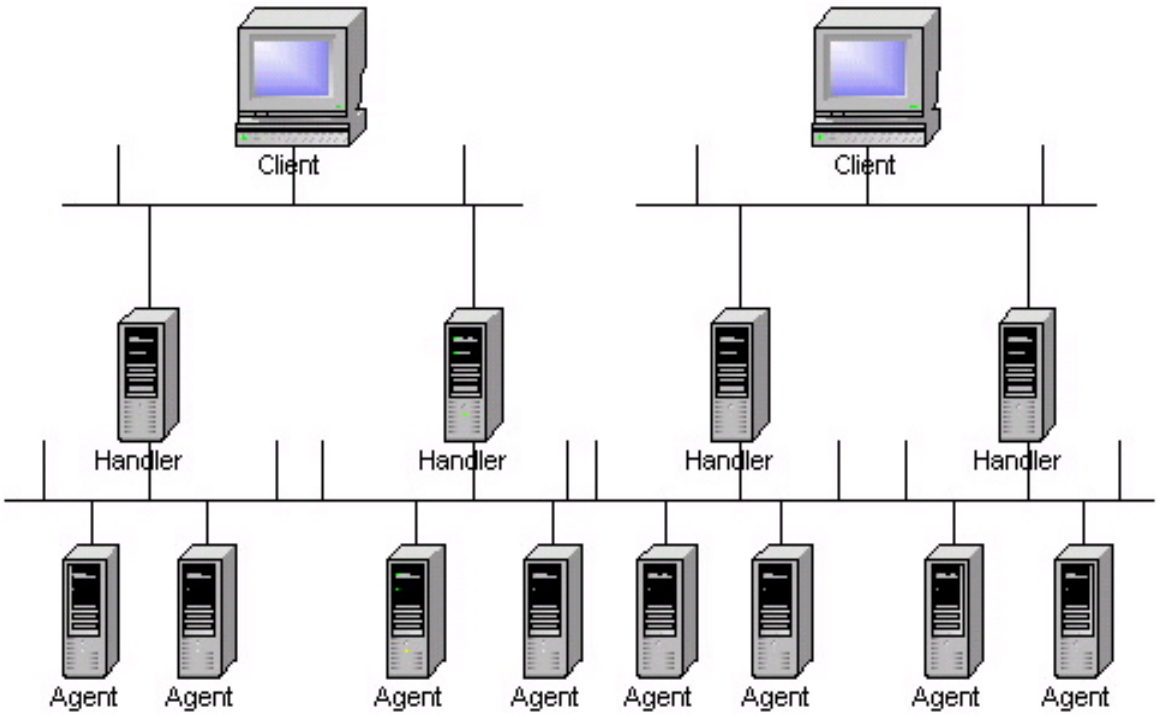
Blocca i broadcasts diretti alla porta 514

```
access-list 110 deny udp any host 0.0.0.0 eq 514
access-list 110 deny udp any host 192.168.0.0 eq 514
access-list 110 deny udp any host 192.168.255.255 eq 514
```

Blocca il traffico verso la porta 514 del router

```
access-list 110 deny udp any host 192.168.1.1 eq 514
```

Distributed Denial of Service



Il **Client** controlla
E attiva l'attacco

Gli **Handler** sono host compromessi che Controllano gli **agents** Schermando I clients

Gli **Agents** sono host compromessi che hanno il compito di realizzare effettivamente gli attacchi

Distributed Denial of Service

Le fasi e la dinamica di un DDoS

- **Scansione di decine di migliaia di hosts per l'individuazione di vulnerabilità note e sfruttabili**
- **Exploit delle vulnerabilità a scopo di compromissione degli host conquistandone l'accesso**
- **Installazione dei tools per la realizzazione del DDoS**
- **Sfruttamento degli hosts conquistati come base di partenza per ulteriori scansioni e compromissioni reiterando il punto 3**
- **Una volta installati i DDoS tools su un numero sufficiente di hosts si procede all'avvio dell'attacco attivando handlers e agents a partire da un client remoto**

Distributed Denial of Service

Caratterizzazione e tipologie

Esiste un certo numero di DDoS tools caratterizzati dalle tecniche di distribuzione dell'attacco fra clients, agent, handlers, dalle porte di default (che possono comunque variare) e dai meccanismi usati per la loro comunicazione

Trinoo	**1524 tcp** **27665 tcp** **27444 udp** **31335 udp**
TFN	**ICMP ECHO/ICMP ECHO REPLY**
Stacheldraht	**16660 tcp** **65000 tcp** **ICMP ECHO/ICMP ECHO REPLY**
TFN2K	**Specificata a runtime o scelta random come combinazione di pacchetti UDP, ICMP and TCP**

Tutte le tecniche in questione realizzano replicatamente attacchi DoS classici (ICMP Bombing, SYN-Flood, Smurfing etc)

Distributed Denial of Service

Tecniche di difesa e contromisure

- Abilitazione di CEF e Unicast Reverse path Forwarding

```
ip verify unicast reverse-path
```

- Applicazione dei filtri anti-spoofing in ingresso e in uscita

```
access-list 110 deny ip 165.21.0.0  0.0.255.255   any log
access-list 110 permit ip any any
access-list 111 permit ip 165.21.0.0 0.0.255.255 any
access-list 111 deny ip any any log
```

- Limitazione in banda dei flussi di traffico ICMP e relativi ai SYN

```
access-list 102 permit icmp any any
access-list 103 deny tcp any any established
access-list 103 permit tcp any any
interface Serial3/0/0
        rate-limit input access-group 102 256000 8000 8000
conform-action transmit exceed-action drop
        rate-limit input access-group 103 256000 8000 8000
conform-action transmit exceed-action drop
```

Caratterizzazione DoS via ACL

In assenza di un'analizzatore di protocollo o di uno sniffer è ugualmente possibile individuare e caratterizzare i principali attacchi di tipo DoS in corso attraverso l'analisi dei "firing counters" di un ACL "di servizio" opportunamente costruita allo scopo:

```
access-list 169 permit icmp any any echo
access-list 169 permit icmp any any echo-reply log-input
access-list 169 permit udp any any eq echo
access-list 169 permit udp any eq echo any
access-list 169 permit tcp any any established
access-list 169 permit tcp any any
access-list 169 permit ip any any
```

```
# show access-list 169
Extended IP access list 169
permit icmp any any echo (2 matches)
permit icmp any any echo-reply (21374 matches)
permit udp any any eq echo
permit udp any eq echo any
permit tcp any any established (150 matches)
permit tcp any any (15 matches)
permit ip any any (45 matches)
```

Caratterizzazione DoS via ACL

Smurfing: Vittima

Il numero di echo-reply ricevuti è elevatissimo rispetto a quello dei request

```
# show access-list 169
…
permit icmp any any echo (2145 matches)
permit icmp any any echo-reply (213746421 matches)
…
```

Gli indirizzi sorgente degli echo reply sono raggruppabili in un insieme limitato di origini che individuano gli amplificatori o “reflectors”

```
# show log
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.212.72
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.212.72
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
```

Caratterizzazione DoS via ACL

Smurfing: Amplificatore

Il numero di echo-request ricevuti è elevatissimo rispetto a quello dei reply

```
# show access-list 169
permit icmp any any echo (214576534 matches)
permit icmp any any echo-reply (4642 matches)
```

Gli indirizzi di destinazione degli echo request individuano dei broadcast diretti ed in genere riportano come sorgente sempre lo stesso indirizzo

```
# show log
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.255 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied icmp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.255 (0/0), 1 packet
```

Si riscontra un elevato numero di broadcast sulla LAN interna

```
# show int fast 4/0/0
FastEthernet4/0/0 is up, line protocol is up
…
     442344667 packets input, 3565139278 bytes, 0 no buffer
     Received 1247787654 broadcasts, 0 runts, 0 giants, …
```

Caratterizzazione DoS via ACL

Fraggle: Vittima

Il numero di udp echo-reply ricevuti è elevatissimo rispetto a quello dei request

```
# show access-list 169
…
permit udp any any eq echo (9845 matches)
permit udp any eq echo any (1374421 matches)
…
```

Gli indirizzi sorgente degli echo reply sono raggruppabili in un insieme limitato di origini che individuano gli amplificatori o “reflectors”

```
# show log
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.45.142
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.212.72
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.212.72
(Serial0 *HDLC*) -> 16.2.3.7 (0/0), 1 packet
```

Caratterizzazione DoS via ACL

Fraggle: Amplificatore

Il numero di udp echo-request ricevuti è elevatissimo rispetto a quello dei reply

```
# show access-list 169
permit udp any any eq echo (45653 matches)
permit udp any eq echo any (64 matches)
```

Gli indirizzi di destinazione degli echo request individuano dei broadcast diretti ed in genere riportano come sorgente sempre lo stesso indirizzo

```
# show log
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.45.142
(Serial0 *HDLC*) -> 10.2.3.255 (0/0), 1 packet
%SEC-6-IPACCESSLOGDP: list 169 denied udp 192.168.45.142
(Serial0 *HDLC*) -> 10.2.3.255 (0/0), 1 packet
```

Si riscontra un elevato numero di broadcast sulla LAN interna

```
# show ip traffic
IP statistics:
…
Bcast: 1147598643 received, 65765 sent
  Mcast: 188967 received, 459190 sent
```

Caratterizzazione DoS via ACL

SYN Flood

Il numero di pacchetti relativi alla fase di 3-way handshake (seconda linea) supera abbondantemente quello di pacchetti su connessioni già stabilite

```
# show access-list 169
…
permit tcp any any established (150 matches)  [socket stabilite]
permit tcp any any (3654 matches)             [socket in syn]
…
```

È inoltre possibile constatare dall'output delcomando **`show log`** la presenza di indirizzi sorgente non validi, oggetto di spoofing.

Ping Flood

Il numero di echo-request e reply ricevuti è elevato con i request che in genere superano i reply. Gli indirizzi sorgente non sono oggetto di spoofing.

```
# show access-list 169
…
permit icmp any any echo (214576534 matches)
permit icmp any any echo-reply (4642 matches)
…
```

Individuazione DoS via netflow cache

E’ possibile individuare la presenza e gli estremi di un DoS in atto attraverso l’analisi della netflow cache riscontrando flussi anomali di traffico che si discostano in maniera evidente dal modello di baseline

```
#show ip cache flow
…
SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP  Pkts
Fa4/0/0       192.132.34.17   AT1/0/0.1     148.240.104.176 06 080C 1388     1
Fa4/0/0       192.132.34.17   AT1/0/0.1     63.34.210.22    06 0AEB 0666    15K
Fa4/0/0       192.133.28.1    Fa4/0/0       143.225.219.187 11 0035 9F37     1
Fa4/0/0       192.132.34.17   AT1/0/0.1     216.207.62.22   06 0FD2 0578  7195
Fa4/0/0       143.225.231.7   AT1/0/0.1     143.225.255.255 11 007F 007D     1
Fa4/0/0       192.132.34.17   AT1/0/0.1     148.240.104.176 06 0015 1381    13
Fa4/0/0       192.132.34.17   AT1/0/0.1     148.240.104.176 06 0015 1382    12
Fa4/0/0       192.133.28.7    AT1/0/0.1     164.124.101.44  11 0035 0035     2
Fa4/0/0       143.225.209.72  AT1/0/0.1     209.178.128.121 01 0000 0000   561K
Fa4/0/0       192.133.28.7    AT1/0/0.1     192.5.5.242     11 0035 0682     1
Fa4/0/0       192.133.28.1    AT1/0/0.1     198.41.0.4      11 0444 0035     1
Se6/7         156.14.1.122    AT1/0/0.1     130.186.1.53    11 0035 0035     1
Fa4/0/0       192.132.34.17   AT1/0/0.1     61.159.200.203  06 0553 042F    75
Fa4/0/0       192.132.34.17   AT1/0/0.1     61.159.200.203  06 052C 0428    12K
…
```

Origine

Destinazione

Traffico