Impianti di elaborazione
AA. 2007-2008

Homework 2 – Hijacking

Patrizio Boschi (278730)
Matteo Corea (269123)
Daniele Libutti (269184)

Descrizione dell'esperienza svolta

Topologia

Il laboratorio realizzato consta di 7 Autonomous System che seguono una struttura gerarchica a tre livelli:

- livello 1: backbone – livello composto da un solo AS (AS1).
- livello 2: provider – livello composto da tre AS (AS10, AS20, AS30). Tra AS20 e AS30 esiste un peering da utilizzare come backup in caso di mancato collegamento diretto con il backbone
- livello 3: customer – livello composto da tre AS (AS100, AS200, AS300). AS200 è il customer collegato a tutti e tre i provider (come da specifiche). AS100 e AS300 sono stub AS, rispettivamente customer di AS10 e AS30.

Gestione degli indirizzi

Tutti i collegamenti punto-punto interdominio sono stati realizzati sfruttando reti /30, numerate progressivamente a partire da 11.0.0.0. Ai provider AS10, AS20 e AS30 sono state assegnate rispettivamente le reti 40.0.0.0/24, 50.0.0.0/24 e 60.0.0.0/24. Ai customer sono state assegnate reti /16: 110.1.0.0/16 (AS100), 120.2.0.0/16 (AS200), 130.3.0.0/16 (AS300). Per creare scenari interessanti sul tema dell'hijacking sono state adottate politiche differenti per la gestione di IP da parte dei diversi customer:

- AS100 – l'intera rete è utilizzata da una singola lan
- AS200 – la /16 è divisa in due reti /17 (politiche di load balancing)
- AS300 – sono utilizzate solo porzioni della /16 assegnata (una rete /17 e una /18). AS300 ha una struttura interna composta da tre router, tra cui sono realizzati collegamenti diretti con indirizzi ricavati da porzioni inutilizzate dell'intervallo inizialmente assegnatogli.

AS1 (backbone) non contiene reti interne.

Connettività

Sono stati implementati alcuni meccanismi di connettività avanzata: AS300 è un multihomed stub AS, con politica di backup (link I è primario, link J è backup). AS200 implementa politiche di loadsharing e backup: di default 120.2.0.0/17 è servita tramite il link F, 120.2.128.0/17 è servita tramite il link G. Il link H è di backup, pronto a prendere in carico il traffico del link F o del link G, in caso di fault di una delle due linee. Ognuno dei tre i link (F, G ed H) è inoltre veicolo dell'intero traffico destinato ad AS200, in caso di fault delle altre due linee. Tale meccanismo è implementato con il seguente sistema: sul link F e sul link G vengono annunciate 4 net /18 (2 per ogni link), in modo da coprire tutto l'intervallo; su tali link viene anche annunciata l'intera /16 (per coprire il fault di due linee). Sul link H vengono annunciate due net /17. In questo modo l'intero intervallo viene potenzialmente servito da tutte le linee, e vengono stabilite priorità di servizio in base al best prefix match. In uscita il traffico da AS200 viene instradato al più vicino gateway di frontiera.
Al fine di modellare una internet realistica, le net /30 non vengono annunciate tramite BGP. Per questo motivo è necessario specificare il source IP per ogni traceroute effettuato a partire da router di frontiera (il traceroute è possibile solo tra ip interni agli AS).
Come da specifiche la default route non viene trasmessa gerarchicamente nella zona rappresentata da backbone e provider (default free zone), ma viene comunque trasmessa dai provider ai customer. Sui customer sono quindi stati implementati filtri degli annunci BGP in ingresso (solo la default route viene accettata) così da non permettere transito di pacchetti tra diversi provider. Tutti gli annunci sono liberi di circolare nella default free zone (nessun filtro implementato sui provider o sulla backbone): questa è una condizione necessaria alla propagazione degli annunci di hijacking.

Testing

Sono stati effettuati test di connettività e test relativi a tecniche di hijacking:

- *connettività*: tramite diversi traceroute è stato testato il corretto funzionamento del laboratorio, in caso di assenza di anomalie e in caso di fault (su diverse linee).
- *hijacking*: sono stati realizzati attacchi hijacking con diverse finalità: address stealing, DoS, sniffing. Per ogni test è stato verificato il comportamento della rete attraverso l'esame delle RIB e FIB di diversi router (il router più interessante è stato senz'altro as1r1), traceroute (con diverse sorgenti e diverse destinazioni) e sniffing di pacchetti su vari link, per verificare l'effettivo traffico in transito.

Considerazioni sui test

I test effettuati mostrano con chiarezza la debolezza di BGP e di internet, a meno di opportuni filtraggi: i filtri agli annunci interdominio rappresentano infatti le barriere che limitano la propagazione dei prefissi annunciati per attuare attacchi hijacking.
Nel modello di internet sviluppato sono state create tutte le condizioni utili a massimizzare i danni degli attacchi (nessun filtro nella default free zone):

- Address stealing: l'attaccante riesce ad appropiarsi di indirizzi ufficialmente appartenenti ad altre organizzazioni. Nella realtà questo potrebbe essere il primo passo per portare a termine azioni illegali (ad esempio spam, o hosting di siti di phishing) e sparire nel nulla dopo un breve periodo di tempo.
- DoS: l'attaccante rende effettivamente irragiungibile un AS vittima (e tutti i servizi da esso erogati). Nella variante implementata l'attacco è indipendente dalla topologia. Annunciando gli stessi prefissi dell'AS vittima si potrebbe realizzare un DoS più localizzato, oscurando la vittima solo a parte della rete (questo scenario potrebbe essere a volte più utile).
- Sniffing: l'attaccante riesce ad appropiarsi del traffico destinato alla vittima, farlo transitare nella propria rete e recapitarlo a destinazione. A nostro avviso questo attacco è il più pericoloso, oltre ad essere di più difficile realizzazione.

Generalmente si è constatato come una tecnica basilare come l'hijacking possa avere un impatto molto forte su tutta la rete.

ID: ***test_connettivita***

SCOPO:

Testare la connettività di base del laboratorio, evidenziando il comportamento di default delle politiche di backup e loadsharing implementate (AS200 e AS300) in condizioni di normale attività della rete (totale assenza di anomalie).

AZIONI DA ESEGUIRE:

1. Traceroute as100r1 => as300r2
   ```
   as100r1:~# traceroute -s 110.1.0.1 130.3.0.1
   ```
2. Traceroute as300r3 => as200r2 (eth1)
   ```
   as300r3:~# traceroute 120.2.128.1
   ```
3. Traceroute as300r3 => as200r2 (eth0)
   ```
   as300r3:~# traceroute 120.2.0.2
   ```
4. Traceroute as1r1 => as200r3 (eth0)
   ```
   as1r1:~# traceroute 120.2.128.2
   ```

RISULTATO ATTESO:

1. Possibilità di raggiungere AS300 da AS100, passando per il backbone (nessun transito in AS200).
2. e 3. Ci si aspetta che il traceroute a partire da as300r3 verso as200r2 segua due strade differenti, a seconda dell'interfaccia da raggiungere, poiché as200r2 è connesso a entrambe le net /17. Per la politica di loadsharing il primo traceroute dovrebbe passare per AS10, il secondo per AS20.
4. Il traceroute a partire dalla radice (as1r1) verso as200r3 dovrebbe passare per AS20 (il link H è di backup)

RISULTATO EFFETTIVO:

1.
```
traceroute to 130.3.0.1 (130.3.0.1) from 110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  1 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  1 ms
 3  11.0.0.9 (11.0.0.9)  4 ms  1 ms  1 ms
 4  60.0.0.2 (60.0.0.2)  2 ms  1 ms  2 ms
 5  11.0.0.34 (11.0.0.34)  2 ms  2 ms  2 ms
 6  130.3.0.1 (130.3.0.1)  2 ms  2 ms  2 ms
```
il traceroute passa per il backbone (riga evidenziata)

2.
```
traceroute to 120.2.128.1 (120.2.128.1), 64 hops max, 40 byte packets
 1  130.3.128.9 (130.3.128.9)  3 ms  1 ms  1 ms
 2  11.0.0.33 (11.0.0.33)  2 ms  1 ms  1 ms
 3  60.0.0.1 (60.0.0.1)  1 ms  1 ms  1 ms
 4  11.0.0.10 (11.0.0.10)  1 ms  2 ms  1 ms
 5  11.0.0.5 (11.0.0.5)  2 ms  2 ms  1 ms
 6  50.0.0.2 (50.0.0.2)  2 ms  2 ms  2 ms
 7  120.2.128.1 (120.2.128.1)  2 ms  2 ms  2 ms
```
sono evidenziati i passaggi in AS20

3.
```
traceroute to 120.2.0.2 (120.2.0.2), 64 hops max, 40 byte packets
 1  130.3.128.9 (130.3.128.9)  1 ms  1 ms  1 ms
 2  11.0.0.33 (11.0.0.33)  1 ms  1 ms  1 ms
 3  60.0.0.1 (60.0.0.1)  1 ms  2 ms  1 ms
 4  11.0.0.10 (11.0.0.10)  3 ms  1 ms  2 ms
 5  11.0.0.1 (11.0.0.1)  2 ms  2 ms  2 ms
 6  11.0.0.22 (11.0.0.22)  2 ms  3 ms  2 ms
 7  120.2.0.2 (120.2.0.2)  2 ms  2 ms  2 ms
```
è evidenziato il passaggio in AS10

4.
```
traceroute to 120.2.128.2 (120.2.128.2), 64 hops max, 40 byte packets
 1  11.0.0.5 (11.0.0.5)  1 ms  1 ms  1 ms
 2  50.0.0.2 (50.0.0.2)  2 ms  1 ms  1 ms
 3  11.0.0.26 (11.0.0.26)  3 ms  2 ms  1 ms
 4  120.2.128.2 (120.2.128.2)  1 ms  1 ms  1 ms
```
anche in questo caso sono evidenziati i passaggi in AS20

test_connettivita ha dato globalmente esito positivo.

ID: ***test_connettivita_avanzata***

SCOPO:

Verificare il corretto funzionamento della rete nei punti di maggiore flessibilità (AS200) in presenza di fault su diverse linee.

AZIONI DA ESEGUIRE:

1. Traceroute as100r1 => as200r2 (eth0), creare fault sul link F, ripetere il traceroute, ripristinare il link F:
   ```
   as100r1:~# traceroute -s 110.1.0.1 120.2.0.2
   as200r1:~# faultF.sh
   ```
   (attendere in modo da far convergere la rete)
   ```
   as100r1:~# traceroute -s 110.1.0.1 120.2.0.2
   as200r1:~# restoreF.sh
   ```
   (attendere in modo da far convergere la rete)
2. Traceroute as100r1 => as200r2 (eth1), creare fault sul link G, ripetere il traceroute, ripristinare il link G:
   ```
   as100r1:~# traceroute -s 110.1.0.1 120.2.128.1
   as200r2:~# faultG.sh
   ```
   (attendere in modo da far convergere la rete)
   ```
   as100r1:~# traceroute -s 110.1.0.1 120.2.128.1
   as200r2:~# restoreG.sh
   ```
   (attendere in modo da far convergere la rete)
3. Creare fault sui link F ed H, quindi traceroute as100r1 => as200r1 (eth1) e traceroute as100r1 => as200r3 (eth0):
   ```
   as200r1:~# faultF.sh
   as200r3:~# faultH.sh
   ```
   (attendere in modo da far convergere la rete)
   ```
   as100r1:~# traceroute -s 110.1.0.1 120.2.0.1
   as100r1:~# traceroute -s 110.1.0.1 120.2.128.2
   as200r1:~# restoreF.sh
   as200r3:~# restoreH.sh
   ```
   (attendere in modo da far convergere la rete)

RISULTATO ATTESO:

1. e 2. ci si aspetta che il traceroute a partire da as100r1 mostri delle variazioni di percorso dovute ai fault sulle linee primarie. In questo caso il traffico della linea interrotta deve passare per il link H.

3. Ci si aspetta che in caso di interruzione di due linee tutto il traffico destinato ad AS200 (i due traceroute servono a verificare il comportamento su tutto l'intervallo di indirizzi) passi per il terzo link, rimasto integro (nell'esempio il link G).

RISULTATO EFFETTIVO:

1.

```
traceroute to 120.2.0.2 (120.2.0.2) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  3 ms  1 ms  1 ms
 2  11.0.0.22 (11.0.0.22)  1 ms  1 ms  1 ms
 3  120.2.0.2 (120.2.0.2)  2 ms  1 ms  2 ms
```

```
traceroute to 120.2.0.2 (120.2.0.2) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  1 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  3 ms  1 ms  1 ms
 3  11.0.0.9 (11.0.0.9)  1 ms  1 ms  1 ms
 4  11.0.0.30 (11.0.0.30)  1 ms  1 ms  2 ms
 5  120.2.0.2 (120.2.0.2)  2 ms  1 ms  2 ms
```

la parte di sinistra riporta l'output del traceroute in assenza di anomalie, la parte destra l'output del traceroute dopo il fault del link F. E' stato evidenziato il passaggio sul link H.

2.

```
traceroute to 120.2.128.1 (120.2.128.1) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  1 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  1 ms
 3  11.0.0.5 (11.0.0.5)  4 ms  1 ms  1 ms
 4  50.0.0.2 (50.0.0.2)  1 ms  2 ms  1 ms
 5  120.2.128.1 (120.2.128.1)  2 ms  2 ms  2 ms
```

```
traceroute to 120.2.128.1 (120.2.128.1) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  1 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  3 ms
 3  11.0.0.9 (11.0.0.9)  1 ms  1 ms  1 ms
 4  11.0.0.30 (11.0.0.30)  1 ms  1 ms  1 ms
 5  120.2.128.1 (120.2.128.1)  2 ms  2 ms  1 ms
```

la parte di sinistra riporta l'output del traceroute in assenza di anomalie, la parte destra l'output del traceroute dopo il fault del link G. E' stato evidenziato il passaggio sul link H.

3.

```
traceroute to 120.2.0.1 (120.2.0.1) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  2 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  1 ms
 3  11.0.0.5 (11.0.0.5)  3 ms  1 ms  2 ms
 4  50.0.0.2 (50.0.0.2)  1 ms  1 ms  1 ms
 5  11.0.0.26 (11.0.0.26)  2 ms  2 ms  1 ms
 6  120.2.0.1 (120.2.0.1)  3 ms  2 ms  2 ms
```

```
traceroute to 120.2.128.2 (120.2.128.2) from
110.1.0.1, 64 hops max, 40 byte packets
 1  11.0.0.17 (11.0.0.17)  1 ms  1 ms  3 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  1 ms
 3  11.0.0.5 (11.0.0.5)  1 ms  1 ms  1 ms
 4  50.0.0.2 (50.0.0.2)  1 ms  1 ms  1 ms
 5  11.0.0.26 (11.0.0.26)  2 ms  2 ms  1 ms
 6  120.2.128.2 (120.2.128.2)  2 ms  2 ms  2 ms
```

Nei due traceroute è stato evidenziato il passaggio sul link G.

test_connettivita_avanzata ha dato esito completamente positivo.

ID: ***test_hijacking_stealing***

SCOPO:

Attacco hijacking con finalità di address stealing: l'attaccante annuncia l'intero intervallo di indirizzi assegnato ad un'altra organizzazione, cercando di trasmettere annunci con una netmask meno selettiva rispetto agli annunci dell'organizzazione bersaglio. In questo modo, per best prefix match, il traffico continua ad arrivare regolarmente alla vittima, e i suoi indirizzi non annunciati vengono dirottati verso l'attaccante. Una volta utilizzati indirizzi altrui l'attaccante ritira gli annunci, e sparisce nel nulla.

AZIONI DA ESEGUIRE:

1. Traceroute as200r3 => indirizzo non annunciato di AS300 (ad esempio 130.3.129.1)
   ```
   as200r3:~# traceroute -s 120.2.128.2 130.3.129.1
   ```
2. Trasmettere annunci di hijacking (AS100 annuncia la /16 di AS300), traceroute as200r3 => as300r3 (eth1):
   ```
   as100r1:~# hijack_steal.sh
   ```
   (attendere in modo da far convergere la rete)
   ```
   as200r3:~# traceroute -s 120.2.128.2 130.3.192.1
   ```
3. Traceroute as200r3 => indirizzo non annunciato di AS300 (ad esempio 130.3.129.1), interrompere l'attacco
   ```
   as200r3:~# traceroute -s 120.2.128.2 130.3.129.1
   as100r1:~# no-hijack.sh
   ```
   (attendere in modo da far convergere la rete)
4. Traceroute as200r3 => indirizzo non annunciato di AS300 (ad esempio 130.3.129.1)
   ```
   as200r3:~# traceroute -s 120.2.128.2 130.3.129.1
   ```

RISULTATO ATTESO:

1. Il traceroute iniziale deve mostrare che l'indirizzo non è raggiungibile (fa parte di quelli non annunciati da AS300)

2. Nonostante la diffusione di indirizzi da parte di AS100, le richieste verso indirizzi annunciati da AS300 devono comunque arrivare a destinazione.

3. Il traceroute eseguito nel punto 1. deve ora tentare di arrivare in AS100. L'attaccante non ha configurato route per accettare tale traffico e conseguentemente, nella emulazione, ci si aspetta che il traffico si fermi nel link E

4. Avendo ritirato gli annunci l'attaccante sparisce nel nulla: il traceroute effettuato al punto 3. deve ora dare come esito la non raggiungibilità, in maniera del tutto analoga a quanto avveniva prima dell'attacco (punto 1.)

RISULTATO EFFETTIVO:

```
1.
traceroute to 130.3.129.1 (130.3.129.1) from 120.2.128.2, 64 hops max, 40 byte packets
 1  11.0.0.9 (11.0.0.9)  4 ms !N  2 ms !N  1 ms !N
```

la destinazione risulta non raggiungibile

```
2.
traceroute to 130.3.192.1 (130.3.192.1) from 120.2.128.2, 64 hops max, 40 byte packets
 1  11.0.0.9 (11.0.0.9)  2 ms  1 ms  1 ms
 2  60.0.0.2 (60.0.0.2)  2 ms  1 ms  2 ms
 3  11.0.0.34 (11.0.0.34)  2 ms  2 ms  2 ms
 4  130.3.192.1 (130.3.192.1)  2 ms  2 ms  3 ms
```

nonostante l'attacco in corso, l'indirizzo 130.3.192.1 risulta ancora raggiungibile

```
3.
traceroute to 130.3.129.1 (130.3.129.1) from 120.2.128.2, 64 hops max, 40 byte packets
 1  120.2.128.1 (120.2.128.1)  3 ms  1 ms  1 ms
 2  11.0.0.25 (11.0.0.25)  1 ms  2 ms  1 ms
 3  50.0.0.1 (50.0.0.1)  1 ms  1 ms  1 ms
 4  11.0.0.6 (11.0.0.6)  1 ms  1 ms  1 ms
 5  11.0.0.1 (11.0.0.1)  2 ms  2 ms  1 ms
 6  11.0.0.18 (11.0.0.18)  2 ms  2 ms  2 ms
 7  11.0.0.1 (11.0.0.1)  2 ms  2 ms  2 ms
 8  11.0.0.18 (11.0.0.18)  2 ms  2 ms  3 ms
 9  * 11.0.0.1 (11.0.0.1)  2 ms  2 ms
10  11.0.0.18 (11.0.0.18)  2 ms  2 ms  3 ms
11  11.0.0.1 (11.0.0.1)  2 ms *  2 ms
12  11.0.0.18 (11.0.0.18)  3 ms  2 ms  3 ms
13  11.0.0.1 (11.0.0.1)  3 ms  3 ms
```

I pacchetti cercano di raggiungere AS100, fermandosi nel link E (righe evidenziate)

```
4.
traceroute to 130.3.129.1 (130.3.129.1) from 120.2.128.2, 64 hops max, 40 byte packets
 1  11.0.0.9 (11.0.0.9)  17 ms !N  2 ms !N  1 ms !N
```

L'indirizzo non annunciato da AS300 torna ad essere irragiungibile alla fine dell'attacco

test_hijacking_stealing ha dato esito globalmente positivo.

ID: ***test_hijacking_DoS***

SCOPO:

Attacco hijacking finalizzato al DoS di porzioni di rete: l'attaccante annuncia un intervallo di indirizzi appartenenti ad una organizzazione vittima, cercando di trasmettere annunci con una netmask più selettiva rispetto a quella degli annunci dell'organizzazione bersaglio. Per best prefix match **tutto il traffico** diretto verso la vittima viene dirottato verso l'attaccante. La vittima diventa irraggiungibile a tutta la rete. Una variante all'attacco consiste nell'annunciare indirizzi utilizzando la stessa granularità degli annunci della vittima. In questo modo il DoS viene avvertito solo in parte della rete (la parte più vicina all'attaccante).

AZIONI DA ESEGUIRE:

1. AS300 annuncia due net /17 appartenenti ad AS100, coprendo l'intero intervallo di indirizzi: l'attacco ha inizio.
   Traceroute as200r1 => as100r1(eth1) . Interrompere l'attacco.
   `as300r1:~# hijack_DoS.sh`
   (attendere in modo da far convergere la rete)
   `as200r1:~# traceroute -s 120.2.0.1 110.1.0.1`
   `as300r1:~# no-hijack.sh`
   (attendere in modo da far convergere la rete)

RISULTATO ATTESO:

1. Ci si aspetta che il traffico da as200r1 verso as100r1 venga dirottato verso AS300. I pacchetti si devono fermare sul link I (in caso di assenza di fault sulla rete)

RISULTATO EFFETTIVO:

```
1.
traceroute to 110.1.0.1 (110.1.0.1) from 120.2.0.1, 64 hops max, 40 byte packets
 1  11.0.0.21 (11.0.0.21)  11 ms  1 ms  1 ms
 2  11.0.0.2 (11.0.0.2)  1 ms  1 ms  1 ms
 3  11.0.0.9 (11.0.0.9)  1 ms  1 ms  1 ms
 4  60.0.0.2 (60.0.0.2)  2 ms  2 ms  1 ms
 5  11.0.0.34 (11.0.0.34)  2 ms  2 ms  2 ms
 6  60.0.0.2 (60.0.0.2)  2 ms  2 ms  2 ms
 7  11.0.0.34 (11.0.0.34)  2 ms  2 ms  2 ms
 8  * 60.0.0.2 (60.0.0.2)  4 ms  4 ms
 9  11.0.0.34 (11.0.0.34)  8 ms  4 ms  3 ms
10  60.0.0.2 (60.0.0.2)  4 ms
```

I pacchetti si fermano sul link I (righe evidenziate): AS100 non è più raggiungibile

test_hijacking_DoS ha dato esito positivo.

ID: ***test_hijacking_sniffing***

SCOPO:

Attacco hijacking finalizzato allo sniffing di alcune porzioni di traffico destinato alla vittima prescelta. L'attaccante effettua annunci del tutto simili a quelli dell'AS vittima, appropriandosi del traffico ad esso destinato e originato dagli AS a lui vicini. Successivamente, con l'aggiunta di rotte statiche create ad hoc, si occupa di recapitare correttamente il traffico. In questo modo l'AS vittima, a meno di analisi approfondite degli AS path, non riesce ad accorgersi di nulla. E' importante sottolineare come questo attacco, contrariamente agli altri mostrati in precedenza, ha un effetto fortemente influenzato dalla topologia: dato un determinato AS attaccante e un determinato AS obiettivo è possibile che lo sniffing avvenga correttamente solo nel caso in cui la distanza tra il punto in cui i pacchetti vengono catturati indebitamente e il punto in cui vengono reinseriti nella rete ("larghezza" dell'AS attaccante) sia maggiore della distanza tra il punto in cui i pacchetti vengono reinseriti in rete e l'origine degli annunci BGP dell'AS vittima (altrimenti i pacchetti reinseriti verrebbero ricatturati dall'attaccante).

AZIONI DA ESEGUIRE:

1. AS200R1 annuncia 130.3.0.0/17, già annunciata da AS300R2.
   ```
   as200r1:~# hijack_sniff.sh
   ```
2. Con alcune rotte statiche si fa transitare il traffico catturato all'interno dell'AS200, ed infine lo si inoltra fuori da esso tramite AS300R3.
   ```
   as200r1:~# route add -net 130.3.0.0 netmask 255.255.128.0 gw 120.2.0.2 dev eth1
   as200r2:~# route add -net 130.3.0.0 netmask 255.255.128.0 gw 120.2.128.2 dev eth1
   as200r3:~# route add -net 130.3.0.0 netmask 255.255.128.0 gw 11.0.0.29 dev eth1
   ```
   (attendere in modo da far convergere la rete)
3. Si effettua uno sniff dei pacchetti su AS200R3
   ```
   as200r3:~# tcpdump -i eth0
   ```
4. Si effettuano dei traceroute a partire da AS100R1 e da AS30R1, entrambi diretti all'AS 300 (la vittima).
   ```
   as100r1:~# traceroute -s 110.1.0.1 130.3.0.1
   as30r1:~# traceroute 130.3.0.1
   ```

RISULTATO ATTESO:

Ci si aspetta che il traffico originato da AS100 e diretto all'AS300 venga intercettato dallo sniffer dell'attaccante (AS200). Il traffico originato dall'AS30 deve invece risultare immune all'attacco hijack, in quanto esso è posizionato in una zona topologicamente più sicura per l'AS300 (è il suo unico provider!)

RISULTATO EFFETTIVO:

```
1. 2. 3. e 4.
 1  11.0.0.17 (11.0.0.17)  6 ms  7 ms  1 ms
 2  11.0.0.22 (11.0.0.22)  2 ms  1 ms  2 ms
 3  11.0.0.26 (11.0.0.26)  3 ms  2 ms  2 ms
 4  11.0.0.30 (11.0.0.30)  1 ms  1 ms  2 ms
 5  11.0.0.9 (11.0.0.9)  1 ms  1 ms  1 ms
 6  60.0.0.2 (60.0.0.2)  2 ms  2 ms  2 ms
 7  11.0.0.34 (11.0.0.34)  2 ms  2 ms  2 ms
 8  130.3.0.1 (130.3.0.1)  2 ms  3 ms  2 ms

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
17:33:25.061829 IP 110.1.0.1.33034 > 130.3.0.1.33441: UDP, length: 12
17:33:25.072976 IP 110.1.0.1.33034 > 130.3.0.1.33442: UDP, length: 12
17:33:25.075905 IP 110.1.0.1.33034 > 130.3.0.1.33443: UDP, length: 12
17:33:25.078187 IP 110.1.0.1.33034 > 130.3.0.1.33444: UDP, length: 12
17:33:25.085995 IP 110.1.0.1.33034 > 130.3.0.1.33445: UDP, length: 12
17:33:25.087884 IP 110.1.0.1.33034 > 130.3.0.1.33446: UDP, length: 12
17:33:25.090064 IP 110.1.0.1.33034 > 130.3.0.1.33447: UDP, length: 12
17:33:25.094802 IP 110.1.0.1.33034 > 130.3.0.1.33448: UDP, length: 12
17:33:25.100614 IP 110.1.0.1.33034 > 130.3.0.1.33449: UDP, length: 12
17:33:25.103004 IP 110.1.0.1.33034 > 130.3.0.1.33450: UDP, length: 12
17:33:25.108227 IP 110.1.0.1.33034 > 130.3.0.1.33451: UDP, length: 12
17:33:25.110704 IP 110.1.0.1.33034 > 130.3.0.1.33452: UDP, length: 12
17:33:25.118407 IP 110.1.0.1.33034 > 130.3.0.1.33453: UDP, length: 12
17:33:25.126938 IP 110.1.0.1.33034 > 130.3.0.1.33454: UDP, length: 12
17:33:25.133938 IP 110.1.0.1.33034 > 130.3.0.1.33455: UDP, length: 12
17:33:25.137080 IP 110.1.0.1.33034 > 130.3.0.1.33456: UDP, length: 12
17:33:25.144198 IP 110.1.0.1.33034 > 130.3.0.1.33457: UDP, length: 12
17:33:25.150705 IP 110.1.0.1.33034 > 130.3.0.1.33458: UDP, length: 12

18 packets captured
18 packets received by filter
0 packets dropped by kernel
```

Il traceroute mostra il passaggio per AS20 (righe evidenziate). Lo sniffer posto in AS20 cattura i pacchetti

test_hijacking_sniffing ha dato esito globalmente positivo