Network Intrusion Detection

Roberto Cecchini

II Incontro di GARR-B
Napoli, 17-18 Gennaio 2000

Mini bibliografia

- M. Crosbie et al., *Intrusion Detection Systems*, COAST Laboratory, Purdue University
  http://www.cerdias.purdue.edu/coast/intrusion-detection/ids.html
- *Michael Sobirey's Intrusion Detection Systems page*
  http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html
- M.J. Ranum, *Intrusion Detection: Challenges and Myths*
  http://www.nfr.com/forum/publications/id-myths.html
- T.H. Ptacek e T.N. Newsham, *Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection*
  http://snort.safenetworks.com/idspaper.html
- *FAQ: Network Intrusion Detection Systems*
  http://www.robertgraham.com/pubs/network-intrusion-detection.html

Tassonomia (1/2)

- In base al modello
  - *Misuse detection*
    - ricerca di pattern specifici o sequenze di eventi (*signatures*);
      - veloci e con pochi falsi positivi;
      - non possono rivelare quello che non conoscono già: necessitano di continui aggiornamenti;
    - *allarmi antifurto*
      - segnalano tutto quello che non dovrebbe succedere
  - *Anomaly detection*
    - ricerca di deviazioni dall'uso "normale" del sistema
      - utilizzano una serie di "modelli" continuamente aggiornati;
      - hanno bisogno di una fase di apprendimento;
      - per reti complesse l'uso "normale" può essere random

Tassonomia (2/2)

- In base alla collocazione
  - *Network-Based*
    - leggono i pacchetti da un'interfaccia di rete in modo promiscuo;
    - installazione più semplice;
    - si possono esaminare gli header dei pacchetti
    - rilevazione in tempo reale;
    - indipendenti dal sistema operativo.
  - *Host-Based*
    - più facile verificare il successo di un attacco;
    - controllano attività specifiche dei sistemi;
    - non richiedono nuovo hardware;
    - si prestano meglio all'uso in ambienti con switch e/o crittografia.

Elusione

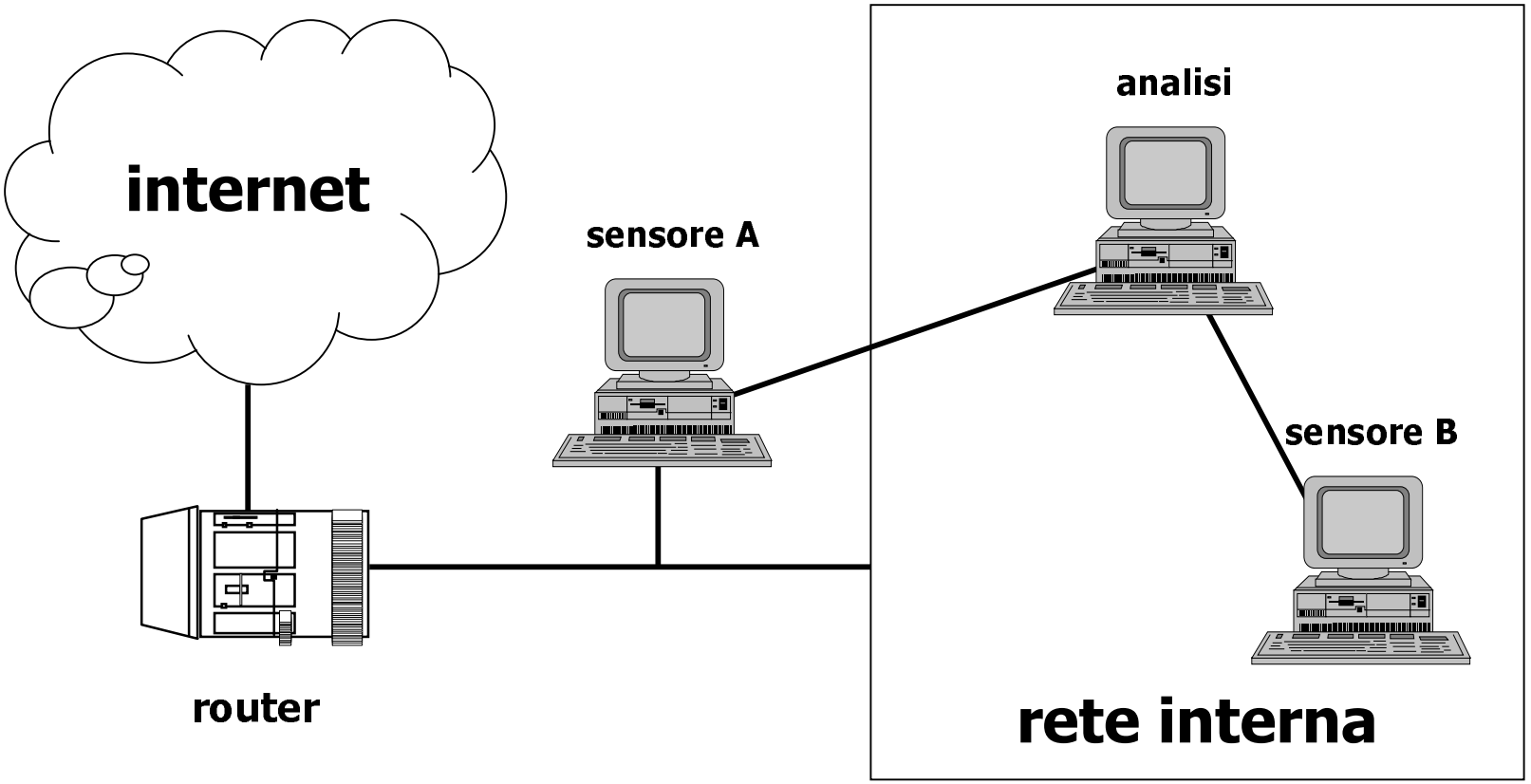
- Un NIDS prevede il comportamento di altre macchine in base ai pacchetti che riceve, ma non tutte sono uguali
  - differenze temporali, pacchetti malformati, condizione degli end-system, ecc.
- Attacchi ai NIDS
  - inserzione
    - pacchetti rifiutati del'end-system, ma accettati dal NIDS
  - evasione
    - pacchetti rifiutati del NIDS, ma accettati dall'end-system
    - scansioni lente
    - attacchi coordinati
  - denial of service
    - CPU: ad es. ricostruzione frammenti;
    - spazio disco;
    - banda passante;
    - contromisure del NIDS (se previste).

NIDS casalingo

- Hardware:
  - almeno una WS unix con almeno 10 GB di spazio disco.
- Software (tutto di pubblico dominio):
  - **libpcap** e **tcpdump**
    - http://www-nrg.ee.lbl.gov/
  - **snort**
    - http://www.clark.net/~roesch
  - **argus**
    - ftp://coast.cs.purdue.edu/pub/tools/unix/argus
  - **arpwatch**
    - http://www-nrg.ee.lbl.gov/
  - **SHADOW**
    - http://www.nswc.navy.mil/ISSEC/CID/

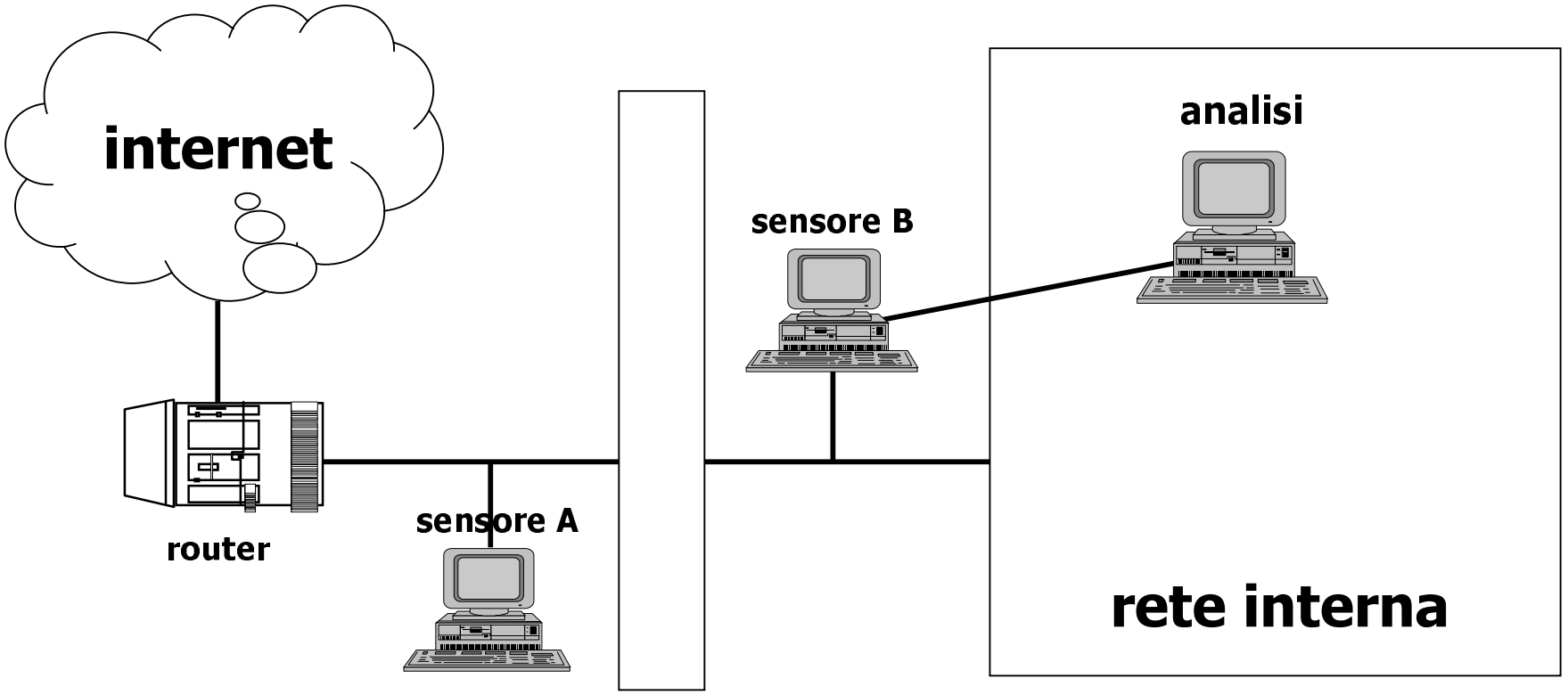
Architettura (semplice)

- la raccolta dati può avvenire in più punti, l'analisi deve essere centralizzata



Architettura (con firewall)

- la raccolta dati può avvenire in più punti, l'analisi deve essere centralizzata



tcpdump

- Connessione tcp (**ssh**)

```
16:14:36.775447 A.1023 > B.ssh: S 1115347652:1115347652(0) win 512 <mss 1460>
16:14:36.775537 B.ssh > A.1023: S 4236737639:4236737639(0) ack 1115347653
    win 17520 <mss 1460> (DF)
16:14:36.775701 A.1023 > B.ssh: . ack 1 win 32120 (DF)
16:14:36.779741 B.ssh > A.1023: P 1:16(15) ack 1 win 17520 (DF) [tos 0x10]
16:14:36.781199 A.1023 > B.ssh: P 1:16(15) ack 16 win 32120 (DF) [tos 0x10]
16:14:36.782557 B.ssh > A.1023: P 16:292(276) ack 16 win 17520 (DF) [tos 0x10]
16:14:36.793025 A.1023 > B.ssh: . ack 292 win 32120 (DF) [tos 0x10]
   ….
16:14:46.818137 A.1023 > B.ssh: P 536:556(20) ack 2324 win 32120 (DF) [tos 0x10]
16:14:46.836499 B.ssh > A.1023: P 2324:2352(28) ack 556 win 17520 (DF) [tos 0x10]
16:14:46.837545 B.ssh > A.1023: P 2352:2372(20) ack 556 win 17520 (DF) [tos 0x10]
16:14:46.837790 A.1023 > B.ssh: P 556:568(12) ack 2372 win 32120 (DF) [tos 0x10]
16:14:46.838312 A.1023 > B.ssh: F 568:568(0) ack 2372 win 32120 [tos 0x10]
16:14:46.838346 B.ssh > A.1023: . ack 569 win 17520 (DF) [tos 0x10]
16:14:46.838443 B.ssh > A.1023: F 2372:2372(0) ack 569 win 17520 (DF) [tos 0x10]
16:14:46.838553 A.1023 > B.ssh: . ack 2373 win 32120 (DF) [tos 0x10]
```

snort: caratteristiche

- Multipiattaforma (Linux, Solaris, *BSD, IRIX, HP-UX)
- Veloce (bassa probabilità di perdere pacchetti)
- Usa un database di regole
- Legge (e scrive) log nel formato di tcpdump
- Accetta filtri di input con la sintassi BPF
- Scrive su syslog, file o winpopup
  - altre azioni possono essere gestite da programmi come **swatch**
- Possibile inserire preprocessori:
  - ip defrag,tcp stream reassembly, portscanning, ecc.

snort: sintassi regole

- Una sola riga
  - header:
    - azione: **alert**, **log** o **pass**
    - protocollo:tcp, **udp**, **icmp**
    - nodo/porta (**any**, numero o range) origine e destinazione
  - opzioni:
    - flag tcp
    - messaggio
    - contenuto (offset e depth)
    - icmp: tipo e codice
    - TTL
    - ecc.
- Ad esempio:
  alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 any (flags: SF; msg: "S-F Scan";)

snort: esempi di regole (1/2)

- Database regole
  - http://www.whitehats.com/ e http://snort.rapidnet.com/
  - Scansioni
    alert tcp $EXT any -> $INT any (msg: "SYN FIN Scan"; flags: SF;)
    alert tcp $EXT any -> $INT any (msg: "NULL Scan"; flags: 0;)
    alert tcp $EXT any -> $INT any (msg: "XMAS Scan"; flags: FPU;)
  - Backdoor
    alert udp $EXT any -> $INT 31335 (msg: "trin00: dmn-mst"; content: "*HELLO*";)
    alert udp $EXT any -> $INT 27444 (msg: "trin00: mst-dmn"; content: "l44adsl";)
  - Buffer overflow
    alert tcp $EXT any -> $INT 110 (msg:"QPOP";flags: PA; content:"|E8 D9FF FFFF|/bin/sh";)
    alert tcp $EXT any -> $INT 143 (msg:IMAP-x86";flags: PA; content:"|e8c0ffffff|/bin/sh";)
    alert tcp $EXT any -> $INT 143 (msg:"imapd6";flags:PA; content:"|eb385e89f389d8804601208046|";)
    alert tcp $EXT any -> $INT 143 (msg:"IMAP";flags: PA; content:"|E8 C0FF FFFF|/bin/sh";)
    alert tcp $EXT any -> $INT 143 (msg:"imap1";flags: PA; content:"|e8 c0ff ffff|/bin/sh";)
  - Traceroute
    alert icmp $EXT any -> $INT any (msg: "Traceroute icmp"; ttl: 1; itype: 8;)
  - Accessi a porte chiuse o indirizzi non usati
    alert tcp $EXT any -> $INT 100:600 (flags: S; msg: "TRAP!";)

snort: esempi di regole (2/2)

- Accessi da/ad host "sospetti"

```
alert tcp CATTIVO any <> $INT any  (msg:"connessionetcp TCP da CATTIVO!";)
```

- Controllo di host importanti

```
preprocessor http_decode: 80 443 8080
preprocessor minfrag: 128

pass tcp any any    <>  any 113  # auth
pass udp any any    <>  any 123  # ntp
pass icmp any any   ->  any any  (itype:  0;)     # echo reply
pass icmp any any   ->  any any  (itype:  3;)     # destination unreachable
pass icmp any any   ->  any any  (itype:  8;)     # echo request
pass icmp any any   ->  any any  (itype: 11;)     # time exceeded

pass tcp any any    ->  $POSTINO 110     # pop3
pass tcp any any    ->  $POSTINO 143     # imap
pass tcp any any    ->  $POSTINO 993     # imaps
pass tcp any any    ->  $POSTINO 995     # pop3s
alert tcp any any   <>  $POSTINO any  (flags: S; msg:"POSTINO: connessionetcp sospetta";)
alert udp any any   <>  $POSTINO any  (msg:"POSTINO: connessione UDP sospetta";)
alert icmp any any  <>  $POSTINO any  (msg:"POSTINO: connessione ICMP sospetta";)

pass udp any 53     <>  $WEBSERVER any # DNS
pass udp any any    <>  $WEBSERVER 137 # netbios-ns
alerttcp any any     ->  $WEBSERVER !80  (flags: S; msg:"WEBSERVER: connessione tcp sospetta";)
alert udp any any   <>  $WEBSERVER any  (msg:"WEBSERVER: Connessione UDP sospetta";)
alert icmp any any  <>  $WEBSERVER any  (msg:"WEBSERVER: Connessione ICMP sospetta";)
```

snort: esempio di output

- allarmi

```
[**] FIN Scan [**]
01/13-11:51:40.341942 CATTIVO:51003 -> VITTIMA:22
TCP TTL:47 TOS:0x0 ID:37904
*F**** Seq: 0x0   Ack: 0x0   Win: 0x1000

[**] Tiny Fragments - Possible Hostile Activity [**]
01/13-11:52:21.723762 CATTIVO -> VITTIMA
TCP TTL:56 TOS:0x0 ID:58930  MF
Frag Offset: 0x0   Frag Size: 0x1A
```

- pacchetti

```
 01/13-11:52:15.720579 CATTIVO -> VITTIMA
TCP TTL:56 TOS:0x0 ID:64046  MF
Frag Offset: 0x0   Frag Size: 0x1A
8A F9 00 16 D2 B3 E8 B4 00 00 00 00 50 01 08 00  ............P...
08 00 08 00 08 00 08 00 08 00                    ..........

01/13-11:52:21.723762 CATTIVO -> VITTIMA
TCP TTL:56 TOS:0x0 ID:58930  MF
Frag Offset: 0x0   Frag Size: 0x1A
8A FA 00 16 07 77 6F 44 00 00 00 00 50 01 08 00  .....woD....P...
08 00 08 00 08 00 08 00 08 00                    ..........
```

argus

- **argus** (server)
  - ascolta dall'interfaccia di rete (o legge un file di **tcpdump**)
  - scrive su file o su socket
- clienti (**ra**, **adjacency**, **racompress**, **dnsstats**, ecc.)
  - leggono i file prodotti dal server (o da socket)
  - producono dati sulle connessioni
- si possono scrivere clienti od eseguire elaborazioni specifiche sui dati
  - scansioni lente
  - network mapping
  - ecc.

argus: esempio d'uso (1/2)

- elenco accessi ad una macchina violata (VITTIMA)
  - **ra -r argus.991212 -c host VITTIMA**

```
12/12 20:49:36 *  tcp  RETEVISION.1031   ->      VITTIMA.23    1550  1505   56659   41352   CLO
12/12 20:50:51    tcp      VITTIMA.1067   -> TECHNOTRONIC.21    26    19     141     1332    CLO
12/12 20:51:12    tcp      VITTIMA.1068  <-  TECHNOTRONIC.20     4     5       0     2451    CLO
12/12 21:14:59    tcp      VITTIMA.1071   ->    BARDOLINO.21    20    16     109      410    CLO
12/12 21:15:10    tcp      VITTIMA.1072  <-     BARDOLINO.20     6     9       0     8145    CLO
12/12 21:26:25 *  tcp RETEVISION2.1028   ->      VITTIMA.23    2405  2362    1658   146227   CLO
12/12 21:29:47    tcp      VITTIMA.1080   ->      TITANIA.23    283   239     161    13700   CLO
12/12 21:29:47    tcp      TITANIA.26862  ->      VITTIMA.113     5     5       9       36    CLO
12/12 22:40:11 *  tcp RETEVISION2.1053   ->      VITTIMA.23    4938  5342  458935   90861   CLO
12/12 23:07:27 s  tcp      VITTIMA.1143   o>         IRC1.6667  2     0       0       0     TIM
12/12 23:10:22    tcp      VITTIMA.1147  <|          IRC2.6667  1     1       0       0     RST
12/12 23:10:56 d  tcp      VITTIMA.1152   ->         IRC3.6667  169   179    1367    14136   CLO
12/12 23:10:57    tcp         IRC3.22133  ->      VITTIMA.113     5     5      13       39    CLO
12/12 23:24:23 *  tcp      VITTIMA.1168   |>         IRC4.6667  143   125     884     11665   RST
12/12 23:24:24    tcp         IRC4.4531   ->      VITTIMA.113     7     6      169      38    CLO
```

argus: esempio d'uso (2/2)

- elenco accessi ad una macchina violata (VITTIMA)
  - **adjacency -r argus.991212 tcp and host VITTIMA**

```
Total Nets             23
Total Hosts            32
Total Cons            195      15797.77       3279704.23       60.89

  entity              cons     src bytes        dst bytes       secs

 VITTIMA         dst    66      46058.44          9934.64     102.25
           pt 20        24     126279.04             0.00       9.17
           pt 23        16        464.25         40898.62     362.84
           pt 1179       2        648.00           207.00     161.45
                src    129        315.57       4952609.60      39.73

       62.0.0.0 dst      6        411.33          2905.83     347.12
SGI4106ef0      dst      5        492.80          3416.00     416.21
            pt 6667      5        492.80          3416.00     416.21
                src      7        194.43            85.71     101.35

    xxx.xxx.xxx dst      7        254.71      91052383.29     151.48
bardolino       dst      3        124.00          1205.67      24.70
           pt 23         2        131.50          1603.50      30.08
                src      1       8145.00             0.00       0.08

   yyy.yyy.yyy  dst      2        488.00          5832.50     452.25
 irc            dst      2        488.00          5832.50     452.25
           pt 6667       2        488.00          5832.50     452.25
                src      1        169.00            38.00       1.19
```

arpwatch

- Controlla e segnala le variazioni ARP:

hostname:	**fpatpace**
ip address:	**193.44.45.112**
ethernet address:	**0:a:27:ae:20:be**
ethernet vendor:	**<unknown>**
old ethernet address:	**0:5:2:c2:18:a**
old ethernet vendor:	**Apple (PCI bus Macs)**
timestamp:	**Monday, January 10, 2000 14:17:27 +0100**
previous timestamp:	**Monday, January 10, 2000 14:11:20 +0100**
delta:	**6 minutes**

- Utile per:
  - ARP cache poisoning;
  - tentativi di neutralizzare gli switch;
  - utenti distratti (o indisciplinati…).
- Attenzione se si usa DHCP.

Esempio di output di **SHADOW**

```
06:06:05.932266 63.65.248.67.interintelli > NS-SEC.traceroute: udp 36 [ttl 1]

12:32:15.628544 ax.xxx.xxx.x.1710 > MAILSERVER.telnet: S 308480000:308480000(0) win 6144
    <mss 3072> (DF)

20:53:41.170867 Irnet-One-gw.RoSprint.net > WEBSERVER icmp: 194.84.37.70 unreachable -
    need to frag (mtu 1486) [tos 0x68]

04:32:32.660706 termsyd43.ozemail.com.au > WEBSERVER icmp: source quench

03:33:34.835448 3dns1.ny.periscopio.com.tragic > NS-SEC.traceroute: udp 36 [ttl 1]

04:41:33.152914 209.67.42.162.tragic > NS-SEC.traceroute: udp 36 [ttl 1]

04:45:57.773372 209.67.42.162.tragic > NS-PRI.traceroute: udp 36 [ttl 1]

05:33:20.198685 209.67.42.162.tragic > NS-SEC.traceroute: udp 36 [ttl 1]

03:35:51.445031 209.67.42.163.tragic > NS-PRI.traceroute: udp 36 [ttl 1]

04:35:01.588658 209.67.42.224.tragic > NS-PRI.traceroute: udp 36 [ttl 1]
```