Analisa Network dengan TCPdump

Reza Muhammad
withoutfx@telkom.net

TCP/IP merupakan standar *de facto* untuk komunikasi antara dua komputer atau lebih. IP (*Internet Protocol*) menjalankan fungsinya pada *network layer* (pengalamatan dan routing) sedangkan TCP (*Transmission Control Protocol*) menyediakan jalur hubungan *end-to-end* (*transport layer*).
TCPdump adalah tools yang berfungsi meng*capture*, membaca atau men*dump*ing paket yang sedang ditransmisikan melalui jalur TCP.

TCPdump diciptakan untuk menolong programer ataupun administrator dalam menganalisa dan *troubleshooting* aplikasi *networking*. Seperti pisau yang bermata dua (hal ini sering kali disebut-sebut), TCPdump bisa digunakan untuk bertahan dan juga bisa digunakan untuk menyerang. *Utility* ini juga seringkali digunakan oleh para cracker untuk melaksanakan perkerjaannya, karena TCPdump bisa meng*capture* atau men*sniff* semua paket yang diterima oleh *network interface*,

Sama halnya dengan tujuan diciptakannya TCPdump, dalam artikel ini saya akan coba membahas bagaimana TCPdump digunakan untuk menganalisa koneksi yang terjadi antara dua sistem.

Langkah pertama yang harus anda lakukan adalah menginstall TCPdump pada box anda. Anda bisa mendapatkan TCPdump di http://www.tcpdump.org. TCPdump membutuhkan libpcap (*packet capture library*) yang juga tersedia di situs TCPdump. Libpcap harus diinstal terlebih dahulu dalam mesin anda. Tentunya anda juga membutuhkan *software* lain untuk mengkompile TCPdump dan libpcap seperti gcc.

Ekstrak libpcap:
[root@your.host direktori_anda]# tar xfzv libpcap-0.x.x.tar.gz

Masuk ke direktori ekstrak, **configure**, **make** & **install**:
[root@your.host direktori_anda]# cd libpcap-0.x.x && ./configure && make && make install

Jika anda tidak ingin menginstal libpcap dalam *library* sistem anda (mungkin hanya untuk percobaan), anda masih dapat menggunakan TCPdump asalkan direktori libpcap berada di bawah direktori yang sama dengan direktori TCPdump (lihat file **INSTALL** dari libpcap).

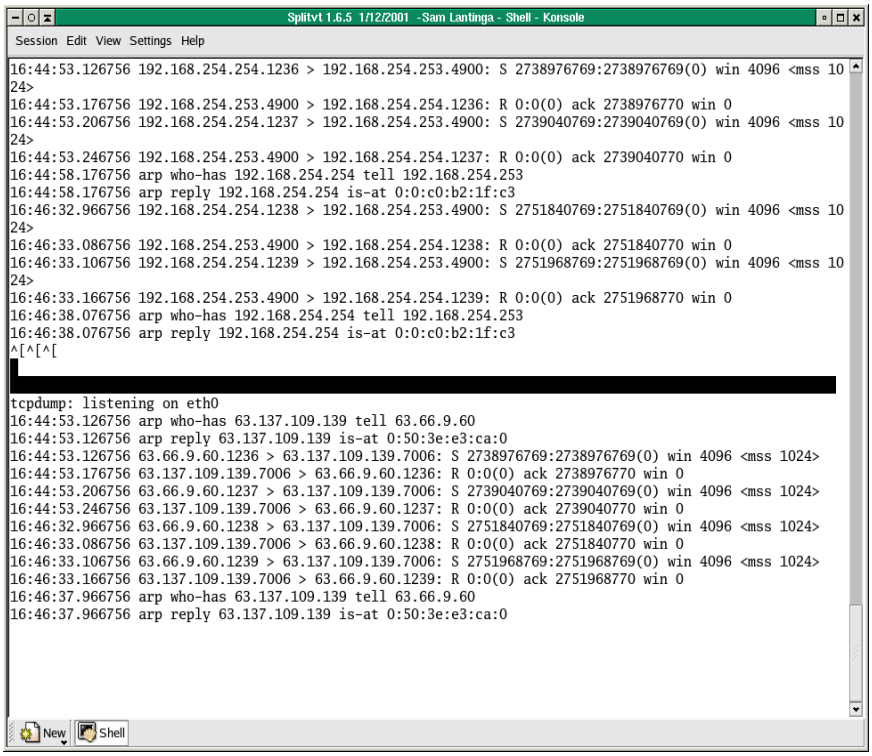
Instalasi TCPdump sama halnya dengan libpcap. cukup ekstrak file-filenya, pergi ke direktori tempat file diekstrak dan jalankan *script* **configure** yang akan menuliskan konfigurasi ke **Makefile.in** lalu jalankan **make** dan **make install**.

Sebagai contoh, saya menggunakan *network* yang terdiri dari tiga komputer yang dihubungkan menggunakan *hub*. Komputer pertama, menggunakan Windows™ 98 dengan IP *address* 192.0.0.1, sedang melakukan koneksi melalui **telnet** ke komputer kedua yang menggunakan Slackware 8.0 dengan IP *address* 192.168.0.2 dan host ketiga komputer Redhat 7.1 dengan IP *address* 192.168.0.3 yang menggunakan *utility* TCPdump. Alasan untuk membedakan *Operating System* yang digunakan adalah untuk menunjukkan bahwa TCP/IP dapat berkomunikasi dengan baik pada dua platform yang berbeda.

Untuk menjalankan TCPdump, ketik perintah **tcpdump** di *console* anda pada *host* 192.168.0.3 sebagai **root**. *Output* yang diperlihatkan di bawah ini adalah *output* yang bergulir *non-stop*, terus berganti baris tanpa henti hingga anda menekan Ctrl+C (^C) untuk menghentikan *utility* ini.

```
# tcpdump
tcpdump: listening on eth0
05:22:27.216338 burner.ssh > prime.1035:
P3797249897:3797249949(52) ack 2183278948 win 8576 (DF) [tos 0x10]
```

Contoh *output* yang berulang diatas menyatakan bahwa salah satu komputer sedang menjalankan **ssh** *client* untuk koneksi ke *server* **ssh** di Redhat 7.1 (192.168.0.3) yang mengakibatkan trafik pada *network* (untuk menghasilkan *output* seperti diatas anda harus menjalankan **ssh** *server* pada Redhat 7.1 dan menggunakan **ssh** *client* untuk melakukan koneksi ke *server* dengan Slackware 8.0 --ini hanya contoh sementara--). Memang seperti yang diharapkan, TCPdump melakukan tugasnya dengan baik. Perintah **tcpdump** bisa digunakan dengan menggunakan beberapa opsi yang dapat menghasilkan *output* yang lebih spesifik. Manual *pages* dari TCPdump menjelaskan opsi-opsi tersebut secara detil.



command line berikut ini akan memulai TCPdump dan memperlihatkan frame (*Protocol Data Unit* (PDU) yang memindahkan data dari pengirim ke tujuan di dalam *network*, menurut **OSI model**, PDU yang berada di *datalink layer* disebut sebagai frame, PDU yang berada di *network layer* disebut paket, dan PDU yang berada di *transport layer* disebut segmen) yang mengandung IP *address* 192.168.0.2 (mesin Slackware)

```
# tcpdump host 192.168.0.2
tcpdump: listening on eth0
19:16:04.817889 arp who-has tssoss tell prime
19:16:04.818025 arp reply tssoss is-at 0:a0:c9:20:5b:fe
19:16:04.818182 prime.1219 > tssoss.telnet:
S2506660519:2506660519(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

Perintah berikut ini hanya akan mengeluarkan *output* pada frame yang mengandung spesifik IP *address* dan spesifik *port*. Opsi **-nn** akan membuat *disable* translasi nama dan *port*.

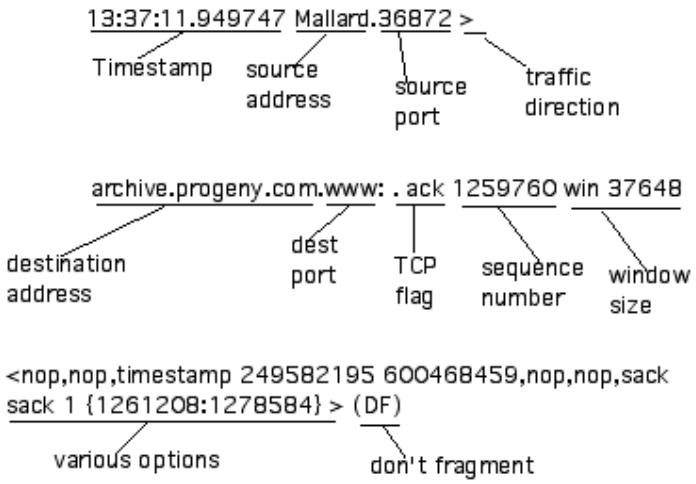
```
# tcpdump -nn host 192.168.0.2 and port 23
tcpdump: listening on eth0
19:20:00.804501 192.168.0.1.1221 > 192.168.0.2.23:
S2565655403:2565655403(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

Jika pada perintah di atas ditambahkan opsi **–t**, *display output* akan ditampilkan tanpa keterangan waktu. Sedangkan penambahan opsi **-e** akan menampilkan **layer** kedua dari informasi *datalink*. Keterangan MAC *address* baik milik mesin pengirim maupun mesin tujuan adalah salah satu contoh dari informasi *datalink*.

```
# tcpdump -nne host 192.168.0.2 and port 23
tcpdump: listening on eth0
19:30:13.024247 0:5:5d:f4:9e:1f 0:a0:c9:20:5b:fe 0800 62: 192.168.0.1.1223 > 192.168.0.2.23:
S2718633695:2718633695(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

Memang banyak sekali *output* yang dikeluarkan oleh TCPdump. Lalu apakah maksud dari *output-output* tersebut ?. *Output* pada TCPdump menampilkan informasi-informasi tentang PDU yang diambil dari frame yang dibaca/di*capture*. Keterangan berikut dapat menjelaskan arti *output* pada contoh diatas (**tcpdump -nn host 192.168.0.2 and port 23**):

Field Contents	**Keterangan**
19:20:00.804501	Deskripsi waktu
192.168.2.10.1221	Alamat IP asal dengan nomor port 1221
192.168.2.165.23	Alamat IP tujuan dengan nomor port 23
S	*flag*/Bendera
2565655403	Nomor urutan data /*data sequence number*
win 16384	*Window size*

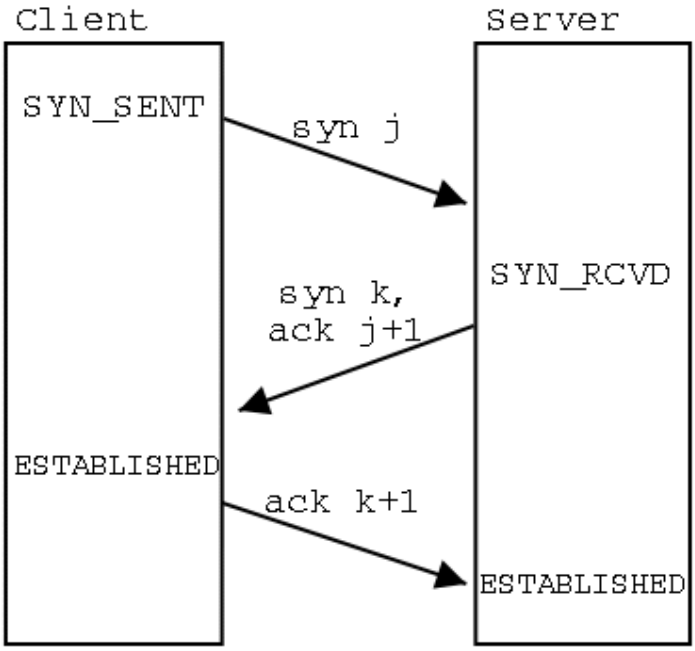


Sebenarnya masih banyak data field yang ditampilkan pada *output*, yang tidak disebutkan pada keterangan diatas. Manual *pages* dari TCPdump mempunyai penjelasan yang cukup tentang *output* yang ditampilkan.

Pemahaman yang baik tentang operasi dan konstruksi dari sebuah protokol sangat dibutuhkan untuk melakukan analisa data. Tapi jangan takut, jika anda merasa *content* artikel ini terlalu berat, kita akan bersama-sama untuk memahaminya :).

TCPdump, *utility* ini juga mempunyai kemampuan untuk menganalisa PDU yang memulai dan mengakhiri suatu koneksi TCP/IP. TCP mempunyai mekanisme khusus untuk membuka dan menutup suatu koneksi. Untuk menjamin bahwa *startup* dan *shutdown* koneksi benar-benar terjadi, TCP menggunakan metode dimana ada tiga pesan yang ditukar, metode ini sering juga disebut *three-way-handshake*.

Berikut adalah urutan untuk memulai (*startup*) suatu koneksi:
1. *Host* peminta koneksi 192.168.0.1 mengirimkan bendera sinkronisasi/*synchronization flag* (SYN) dalam segmen TCP untuk membuat suatu koneksi.
2. *Host* penerima permintaan 192.168.0.2 menerima SYN *flag* dan mengirimkan bendera pernyataan/acknowledgment flag (ACK) kepada *host* peminta koneksi.
3. *Host* peminta koneksi akan menerima ACK flag milik 192.168.0.2 sebagai SYN *flag* dan mengembalikan SYN *flag* yang diterima sebagai ACK *flag* nya sendiri kepada 192.168.0.2



Proses *handshake* yang sama juga terjadi saat menutup (*shutdown*) koneksi, tetapi bendera atau *flag* yang digunakan adalah bendera selesai atau *finish flag* (FIN).

Untuk melakukan koneksi, *host* yang meminta koneksi akan membuat suatu segmen (PDU di *transport layer*) yang mengandung alamat IP dan nomor *port* dari *host* yang dituju. Segmen tersebut mengandung SYN *flag* dan *host* peminta akan menginisialisas urutan nomor/*sequence number*. Sebelum data dikirim, data tersebut akan dipecah menjadi bagian-bagian kecil, dan fungsi *sequence number* adalah untuk membantu perakitan kembali data yang telah dipecah di *host* tujuan. Pada *output* TCPdump berikut, kita dapat melihat data-data tersebut :

#tcpdump -nn host 192.168.0.2 and port 23
20:06:32.845356 192.168.0.1.1249 > 192.168.0.2.23:
S 3263977215:3263977215(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

Host yang diminta akan merespon permintaan dengan mengirimkan SYN *flag* dan *sequence number* miliknya sendiri kepada host peminta, respon ini juga mengandung ACK *flag* (3263977215 (SYN *flag* milik peminta) + 1) untuk menyatakan bahwa permintaan SYN *flag* peminta sudah diterima. Lihat *output* berikut:

20:06:32.845725 192.168.0.2.23 > 192.168.0.1.1249: S
48495364:48495364(0) ack 3263977216 win 32120 <mss 1460,nop,nop,sackOK> (DF)

Kemudian *host* peminta akan merespon SYN *flag* yang dikirim oleh *host* yang diminta dengan cara mengirimkan segmen lain yang mengandung **.** *flag* dan ACK *flag*

20:06:32.845921 192.168.0.1.1249 > 192.168.0.2.23: . ack 1 win 17520
(DF)

Sejauh ini kita telah melihat bahwa ada dua *flag* yaitu S dan **.** yang ditampilkan oleh *output* TCPdump. sebenarnya semua *flag* yang ditampilkan oleh *output* TCPdump ada 5 yaitu:

S: SYN (Synchronize sequence numbers - Connection establishment)
F: FIN (Ending of sending by sender - Connection termination)
R: RST (Reset connection)
P: PSH (Push data)
.: (No flag is set)

Untuk menutup koneksi segment yang mengadung SYN *flag* akan dikirimkan dari *host* 192.168.0.2 ke *host* yang membuka/meminta koneksi, lihat *output* berikut:

20:07:32.916410 192.168.0.2.23 > 192.168.0.1.1249: F 147:147(0) ack
56 win 32120 (DF)

Host 192.168.0.1 akan merespon FIN segmen :

20:07:32.916680 192.168.0.1.1249 > 192.168.0.2.23: . ack 148 win
17374 (DF)

Dan kemudian *host* 192.168.0.1 akan memutuskan koneksi dengan mengirimkan segmen yang mengandung FIN *flag*.

20:07:32.928907 192.168.0.1.1249 > 192.168.0.2.23: F 56:56(0) ack 148
win 17374 (DF)

Host 192.168.0.2 pun meresponnya dengan mengirimkan pernyataan :

20:07:32.929121 192.168.0.2.23 > 192.168.0.1.1249: . ack 57 win 32120
(DF)

Pada skenario yang telah diuraikan diatas kita dapat melihat bahwa TCPdump benar-benar teliti, dalam melakukan analisa. TCPdump juga dapat menganalisa masalah lain seperti melacak terjadinya IP *spoofing* melalui pengenalan MAC address, MITM (*man in the middle attack*) ataupun IP *hijacking*. Mudah-mudahan artikel ini dapat bermanfaat bagi anda untuk menganalisa *network* ataupun dalam pembuatan aplikasi *network*. Kritik dan saran yang membangun sangat saya harapkan karena semua ini adalah proses dan bukanlah suatu hasil, apalagi hasil yang sempurna.