Tcdump Usage (Basic)

```
# Listen on interface eth0
tcpdump -i eth0

# Display IP addresses and port numbers instead of domain
# and service names when capturing packets
tcpdump -n

# Display IP addresses and port numbers instead of domain
# and service names when capturing packets
tcpdump -nn                                    // for Fedora and other derivatives like CentOS

# Don't convert IP addresses and capture any packets
# where the source host is 172.16.3.107
tcpdump -i eth0 -n src host 172.16.3.107

# Don't convert IP addresses and capture any packets
# where the destination host is 172.16.3.107
tcpdump -i eth0 -n dst host 172.16.3.107

# Don't convert IP addresses and capture any packets
# where the source port is 23
tcpdump -i eth0 -n src port 23 172.16.3.107
```

Note: Sanal makinadan src portu 23 olan paket ana makinaya gönderildiğinde ana makinadaki tcpdump ekrana src portu 23 olmayan kayıtlar yansıtacaktır. Bu durum port filtresinin çalışmadığı intibaını uyandıracaktır. Ancak port filtresi çalışıyor. Aynı işlem ana makina - ana makina arasında yapıldığında tcpdump olması gerektiği gibi sadece src portu 23 olan paketleri ekrana basıyor. Önceki işlemde yaşanan problem ise büyük olasılıkla sanal makina ve ana makina arasındaki iletişiminin yapısından kaynaklanıyor olmalı. (Not: Paket oluşturma, gönderme ve tcpdump ile izleme işlemleri dökümanın ilerleyen sayfalarında yapılacaktır)

```
# Don't convert IP addresses and capture any packets
# where the destination port is 53
tcpdump -i eth0 -n dst port 53 172.16.3.107

# Don't convert IP addresses and capture any packets
# which come from specific host and specific port
tcpdump -i eth0 -n "src host 172.16.3.107 and src port 53"          // Quote for multiple filter

# Don't convert IP addresses and capture tcp packets.
tcpdump -i eth0 -n tcp

# Don't convert IP addresses and capture udp packets.
tcpdump -i eth0 -n udp

# Don't convert IP addresses and capture arp packets.
tcpdump -i eth0 -n arp
```

```
# Don't convert IP addresses and capture icmp packets.
tcpdump -i eth0 -n icmp

# Don't convert IP addresses and capture tcp packets
# which come from specific host and specific port
tcpdump -i eth0 "src host 172.16.3.117 and src port 53 and tcp"

# Don't convert IP addresses and don't print out timestamp
tcpdump -i eth0 -n -t

# Don't convert IP addresses, don't print out timestamp,
# and print out as verbose
tcpdump -i eth0 -n -t -v                                        // tcpdump -i eth0 -tnv

# Save packets to a file (.pcap)
tcpdump -i eth0 -w capture.pcap

# Read a file
tcpdump -r capture.pcap

# Output headers of each packet as HEX
tcpdump -X -r capture.pcap

# Output mac addresses of each packet
tcpdump -e -r capture.pcap
```

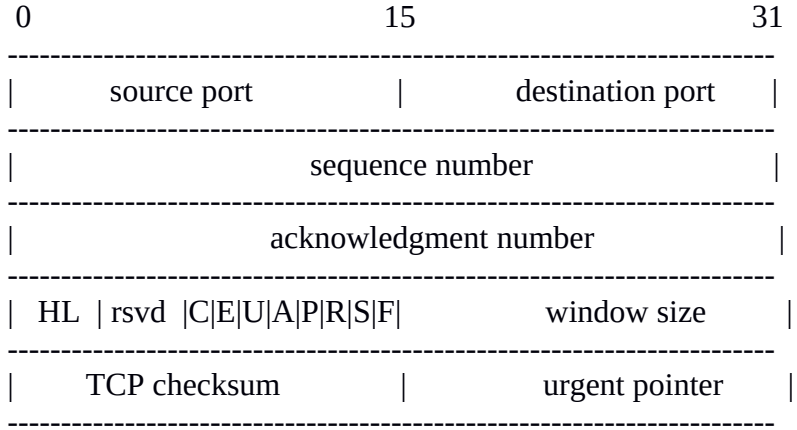
tcpdump TCP packet filter syntax

tcpdump "tcp[tcpflags] == tcpflagNumber"

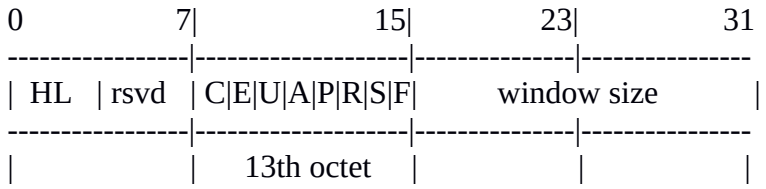
or

tcpdump "tcp[tcpflags] & tcpflag != 0"

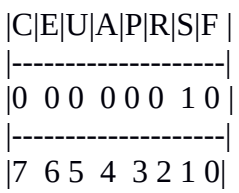
tcpflags : octet number in the tcp header



Flags are in 13th octet in the tcp header.



Let's have a closer look at octet no. 13:



Assuming that octet number 13 is an 8-bit unsigned integer in network byte order, the binary value of this octet is

00000010

and its decimal representation is

7 6 5 4 3 2 1 0
0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 1*2 + 0*2 = 2

Then we write the following command to filter SYN packets:

```
tcpdump  -i  eth0  "tcp[13] == 2"
```

13 : octet number
2 : SYN flag

tcp[13] == 2 expression says take the packets which SYN flag is up in 13th octet.

We can filter syn packets without calculating flag number:

```
tcpdump  -i  eth0  "tcp[13] & tcp-syn != 0"
```

In the above command we use tcp-syn tag instead of bit number of SYN flag. The command above says that take all packets which syn flag isn't zero (i.e. which syn flag is one).

Tcdump TCP Packet Filter Examples

```
# Filter SYN Packets
tcpdump -i eth0 "tcp[13] & tcp-syn != 0"

# Filter ACK Packets
tcpdump -i eth0 "tcp[13] & tcp-ack != 0"

# Filter SYN/ACK Packets
tcpdump -i eth0 "tcp[13] & (tcp-syn & tcp-ack) != 0"

# Filter FIN Packets
tcpdump -i eth0 "tcp[13] & tcp-fin != 0"

# Filter PUSH Packets
tcpdump -i eth0 "tcp[13] & tcp-push != 0"

# Filter RST Packets
tcpdump -i eth0 "tcp[13] & tcp-rst != 0"

# Filter URG Packets
tcpdump -i eth0 "tcp[13] & tcp-urg != 0"

# Filter SYN or ACK packets
tcpdump -i eth0 "tcp[13] & (tcp-syn | tcp-ack) != 0"

# An example of multiple filter
tcpdump -i eth0 "src host 172.16.3.117 and src port 53 and tcp[13] & tcp-syn != 0"
```

hping3 ile Paket Üretimi

Ubuntu IP : 172.16.3.113
Kali Makina IP : 172.16.3.107

i) ICMP Paket Üretimi

Ubuntu

> hping3 --icmp 172.16.3.107

Kali

> tcpdump -i eth0 -t -n icmp

Output:

IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 0, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 0, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 256, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 256, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 512, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 512, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 768, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 768, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 1024, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 1024, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 1280, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 1280, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 1536, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 1536, length 8
IP 172.16.3.113 > 172.16.3.107: **ICMP echo request**, id 45324, seq 1792, length 8
IP 172.16.3.107 > 172.16.3.113: **ICMP echo reply**, id 45324, seq 1792, length 8

ii) SYN Paket Üretimi

Ubuntu

> hping3 --syn 172.16.3.107 // SYN paketleri gönderir.

Kali

> tcpdump -i eth0 -t -n "tcp[13] & tcp-syn != 0"

Output:

IP 172.16.3.113.2746 > 172.16.3.107.0: Flags [**S**], seq 1062084714, win 512, length 0
IP 172.16.3.113.2747 > 172.16.3.107.0: Flags [**S**], seq 1844087843, win 512, length 0
IP 172.16.3.113.2748 > 172.16.3.107.0: Flags [**S**], seq 617315076, win 512, length 0
IP 172.16.3.113.2749 > 172.16.3.107.0: Flags [**S**], seq 1265406527, win 512, length 0
IP 172.16.3.113.2750 > 172.16.3.107.0: Flags [**S**], seq 1675050461, win 512, length 0
IP 172.16.3.113.2751 > 172.16.3.107.0: Flags [**S**], seq 314983707, win 512, length 0
IP 172.16.3.113.2752 > 172.16.3.107.0: Flags [**S**], seq 1160676415, win 512, length 0

IP 172.16.3.113.2753 > 172.16.3.107.0: Flags [S], seq 1086958469, win 512, length 0
IP 172.16.3.113.2754 > 172.16.3.107.0: Flags [S], seq 488145846, win 512, length 0
IP 172.16.3.113.2755 > 172.16.3.107.0: Flags [S], seq 1560952838, win 512, length 0
IP 172.16.3.113.2756 > 172.16.3.107.0: Flags [S], seq 2113414560, win 512, length 0
IP 172.16.3.113.2757 > 172.16.3.107.0: Flags [S], seq 1929615477, win 512, length 0
IP 172.16.3.113.2758 > 172.16.3.107.0: Flags [S], seq 1440115727, win 512, length 0
IP 172.16.3.113.2759 > 172.16.3.107.0: Flags [S], seq 1629263934, win 512, length 0
…

hping3 produces SYN packets continuously and send them to the Kali. Kali listens his packets and filter them with SYN flag. Thereby, tcpdump outputs just SYN packets incoming.

iii) RST Paket Üretimi

Ubuntu

> hping3 --rst 172.16.3.107 // RST paketleri gönderir

Kali

> tcpdump -i eth0 -t -n "tcp[13] & tcp-rst != 0"

Output:

IP 172.16.3.113.2286 > 172.16.3.107.0: Flags [R], seq 1395614016, win 512, length 0
IP 172.16.3.113.2287 > 172.16.3.107.0: Flags [R], seq 1424463796, win 512, length 0
IP 172.16.3.113.2288 > 172.16.3.107.0: Flags [R], seq 197670793, win 512, length 0
IP 172.16.3.113.2289 > 172.16.3.107.0: Flags [R], seq 940310500, win 512, length 0
IP 172.16.3.113.2290 > 172.16.3.107.0: Flags [R], seq 473082393, win 512, length 0
IP 172.16.3.113.2291 > 172.16.3.107.0: Flags [R], seq 2074567585, win 512, length 0
IP 172.16.3.113.2292 > 172.16.3.107.0: Flags [R], seq 180092868, win 512, length 0
IP 172.16.3.113.2293 > 172.16.3.107.0: Flags [R], seq 1486511321, win 512, length 0
IP 172.16.3.113.2294 > 172.16.3.107.0: Flags [R], seq 1796537166, win 512, length 0
IP 172.16.3.113.2295 > 172.16.3.107.0: Flags [R], seq 1532575471, win 512, length 0
IP 172.16.3.113.2296 > 172.16.3.107.0: Flags [R], seq 1244045085, win 512, length 0
IP 172.16.3.113.2297 > 172.16.3.107.0: Flags [R], seq 2000750259, win 512, length 0
IP 172.16.3.113.2298 > 172.16.3.107.0: Flags [R], seq 475735829, win 512, length 0

iv) UDP Paket Üretimi

Ubuntu

> hping3 --udp 172.16.3.107

Kali

> tcpdump -i eth0 -t -n udp

Output:

IP 172.16.3.113.1252 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1253 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1254 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1255 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1256 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1257 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1258 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1259 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1260 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1261 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1262 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1263 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1264 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1265 > 172.16.3.107.0: **UDP**, length 0
IP 172.16.3.113.1266 > 172.16.3.107.0: **UDP**, length 0

Not: Araya tubitak network'ünden gelen başka udp paketleri de girmiştir. Olayı daha sade sunmak için o paketler çıktıdan silinmiştir.

Kaynaklar

http://www.rationallyparanoid.com/articles/tcpdump.html

http://www.tcpdump.org/tcpdump_man.html

http://ask.xmodulo.com/capture-tcp-syn-ack-fin-packets-tcpdump.html

https://superuser.com/questions/587302/how-to-make-tcpdump-to-display-ip-and-port-number-but-not-hostname-and-protocol