IGMP Snooping

man tcpdump | less -Ip examples, чтобы увидеть некоторые примеры

фильтр tcpdump для HTTP GET:

```
sudo tcpdump -s 0 -A 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420'
```

фильтр tcpdump для HTTP POST:

```
sudo tcpdump -s 0 -A 'tcp dst port 80 and (tcp[((tcp[12:1] & 0xf0) >> 2):4]
= 0x504f5354)'
```

Мониторинг HTTP-трафика, включая заголовки запросов и ответов и тело сообщения (источник):

```
tcpdump -A -s 0 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) -
((tcp[12]&0xf0)>>2)) != 0)'
tcpdump -X -s 0 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) -
((tcp[12]&0xf0)>>2)) != 0)'
```

## List interfaces that tcpdump can listen on

```
# tcpdump -D
1.eth0
2.eth1
3.eth1.780
4.eth1.781
5.eth1.790
6.eth2
7.eth2.10
8.eth3
9.eth4
10.any (Pseudo-device that captures on all interfaces)
11.lo
```

Note: «any» interface is an option only on Linux systems running kernel 2.4 onwards. Not available on *BSD, Solaris or any other Unix system.

## Turn on "verbose" key in TCPDUMP to see IP and TCP header information

```
tcpdump -vi eth0
```

## Turn off hostname and port lookup in TCPDUMP

```
tcpdump -vnni eth0
```

## Tcpdump filter only icmp traffic

```
tcpdump -nni eth0 icmp
```

## Tcpdump command to filter on ICMP type - capture only ICMP echo request

As shows on ICMP wiki page http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol, ICMP echo requests are ICMP type 8 ( ICMP code is not important as there is only one code for ICMP type 8 [ and 0 actually ] )

```
tcpdump -nni vlan111 -e icmp[icmptype] == 8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vlan111, link-type EN10MB (Ethernet), capture size 65535 bytes
12:39:05.471531 00:07:e9:a5:9b:fa > 00:10:db:ff:10:02, ethertype IPv4
(0x0800), length 98: 10.1.111.10 > 10.0.0.4: ICMP echo request, id 24907,
seq 307, length 64
12:39:06.472431 00:07:e9:a5:9b:fa > 00:10:db:ff:10:02, ethertype IPv4
(0x0800), length 98: 10.1.111.10 > 10.0.0.4: ICMP echo request, id 24907,
seq 308, length 64
```

Above tcpdump filter «icmp[icmptype] == 8» will only display ip packets that have icmp payload and icmptype 8 - ICMP Echo Request.

## Tcpdump command to filter on ICMP type - capture only ICMP echo reply

```
# tcpdump -nni vlan111 -e icmp[icmptype] == 0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vlan111, link-type EN10MB (Ethernet), capture size 65535 bytes
12:40:52.569668 00:10:db:ff:10:02 > 00:07:e9:a5:9b:fa, ethertype IPv4
(0x0800), length 98: 10.0.0.4 > 10.1.111.10: ICMP echo reply, id 24907, seq
414, length 64
12:40:53.570530 00:10:db:ff:10:02 > 00:07:e9:a5:9b:fa, ethertype IPv4
(0x0800), length 98: 10.0.0.4 > 10.1.111.10: ICMP echo reply, id 24907, seq
415, length 64
```

Notice from ICMP types and codes table that icmptype 0 is the echo reply.

## Tcpdump filter packets with specified ip identification in ip header

(See https://forum.ivorde.com/tcpdump-filter-packets-with-specified-ip-identification-in-ip-header-t19601.html for more details)

```
# tcpdump -nr /tmp/tcpdump.pcap -v 'ip[4:2] == 24332'
reading from file /tmp/tcpdump.pcap, link-type EN10MB (Ethernet)
capability mode sandbox enabled
```

```
23:58:50.090759 IP (tos 0x10, ttl 128, id 24332, offset 0, flags [DF], proto
TCP (6), length 204)
    10.1.1.1.22 > 192.168.0.109.53989: Flags [P.], seq
3661036793:3661036957, ack 2364476704, win 4106, length 164
```

### Tcpdump filtering based on DSCP field in IP header

(See https://forum.ivorde.com/tcpdump-how-to-to-capture-only-ip-packets-with-specific-dscp-class-in-ip-header-t14041.html for more details)

```
# tcpdump -nni eth1 -v 'ip[1] & 0xfc == 184'
12:56:49.690239 IP (tos 0xb8, ttl 63, id 44823, offset 0, flags [DF], proto
TCP (6), length 40)
    28.32.179.11.80 > 61.219.73.106.61244: Flags [F.], cksum 0xdac1
(correct), seq 2799324281, ack 4189664666, win 108, length 0
```

### Tcpdump: How to to capture only ICMP Fragmentation needed notifications

(See https://forum.ivorde.com/tcpdump-how-to-to-capture-only-icmp-fragmentation-needed-notifications-t15211.html)

```
# tcpdump -nni vlan111 -e icmp[icmptype] == 3 && icmp[icmpcode] == 4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vlan111, link-type EN10MB (Ethernet), capture size 65535 bytes
12:46:41.500646 00:10:db:ff:10:02 > 00:07:e9:a5:9b:fa, ethertype IPv4
(0x0800), length 70: 10.1.111.1 > 10.1.111.10: ICMP 10.0.0.3 unreachable -
need to frag (mtu 1382), length 36
```

### How to capture frames with specific source or destination mac address

(See https://forum.ivorde.com/tcpdump-how-to-capture-frames-with-specific-source-destination-mac-address-t19471.html)

```
tcpdump -nni eth0 ether src 2c:21:72:c6:c1:88
```

```
tcpdump -nni eth0 ether dst 2c:21:72:c6:c1:88
```

### Capture only packets from a specific IP host or to a specific IP destination

```
$ tcpdump -nni en0 src host 8.8.8.8.8
```

```
$ tcpdump -nni en0 dst host 8.8.8.8.8
```

## Tcpdump - capture only ARP packets

```
$ tcpdump -nni en0 arp
```

## Capture only IPv4 or only IPv6 traffic

```
$ tcpdump -nni en0 ip
```

```
$ tcpdump -nni en0 ip6
```

## Capture ethernet multicast traffic based on ethernet field and on IPv4 destination

```
$ tcpdump -nni en0 "ether[0] & 1 != 0"
```

(Make sure the tcpdump expression above is enclosed in double quotes otherwise the & will be interpreted by the shell not by tcpdump).

```
$ tcpdump -nni en0 dst net 224.0.0.0/4
```

## Show ethernet / layer 2 headers

```
$ tcpdump -nni en0 -e
21:54:03.017194 80:71:1f:39:61:c8 > 80:e6:50:07:2d:d6, ethertype IPv4
(0x0800), length 126: 64.233.166.189.443 > 192.168.3.100.57904: Flags [P.],
seq 2082387620:2082387680, ack 1352514330, win 1373, options [nop,nop,TS val
1373308623 ecr 829302794], length 60
```

## Capture only specific vlan traffic (for interfaces that terminate vlan trunks)

```
# tcpdump -nni em2 -e vlan 5
20:55:32.265019 f8:c0:01:d2:35:c1 > 00:26:0b:28:5e:40, ethertype 802.1Q
(0x8100), length 370: vlan 5, p 0, ethertype IPv4, 12.16.11.149 >
12.250.3.6: ESP(spi=0x0f3e6725,seq=0x72f), length 332
```

## Capture specific IPv4 protocols related traffic

IPv4 protocols are defined in any Linux/*BSD under /etc/protocols so if your having a temporary lack of memory, it's place to match protocol names against their id.

```
# grep -E "esp|ah|gre|ospf|icmp|tcp|udp" /etc/protocols
icmp    1    ICMP       # internet control message protocol
tcp    6    TCP       # transmission control protocol
```

```
udp    17    UDP       # user datagram protocol
gre    47    GRE       # General Routing Encapsulation
esp    50    IPSEC-ESP    # Encap Security Payload [RFC2406]
ah    51    IPSEC-AH    # Authentication Header [RFC2402]
ipv6-icmp 58    IPv6-ICMP    # ICMP for IPv6
ospf    89    OSPFIGP       # Open Shortest Path First IGP
udplite    136    UDPLite       # UDP-Lite [RFC3828]
wesp    141    WESP       # Wrapped Encapsulating Security Payload
```

Showing below how to capture GRE traffic.

```
# tcpdump -nni em2 ip proto 47
tcpdump: WARNING: em2: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on em2, link-type EN10MB (Ethernet), capture size 65535 bytes
21:17:32.870695 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto GRE
(47), length 100)
    12.16.81.123 > 82.210.283.106: GREv0, Flags [none], length 80
   IP6 (class 0xc0, hlim 1, next-header OSPF (89) payload length: 36)
fe80::fac0:100:d2:3580 > ff02::5: OSPFv3, Hello, length 36
   Router-ID 172.16.2.2, Backbone Area
   Options [V6, External, Router]
     Hello Timer 10s, Dead Timer 40s, Interface-ID 0.0.0.2, Priority 128
     Neighbor List:
```

## Dump HTTP data as ASCII or ASCII and HEX

(See https://forum.ivorde.com/tcpdump-dump-http-headers-as-ascii-and-hex-t19591.html for more details)

```
# tcpdump -nni eth0 -s0 -A -l port 80
```

```
# tcpdump -nni eth0 -s0 -AX -l port 80
```

The output can be filtered with grep to only dump specific attribute in HTTP header or specific html tag inside the payload.

## Capture only traffic related to a CIDR subnet

```
# tcpdump -nni eth0 net 192.168.3.96/28
02:48:33.958798 IP 10.1.22.2.22 > 192.168.3.100.61644: Flags [P.], seq
2001101694:2001101886, ack 4183269133, win 49, options [nop,nop,TS val
1422334877 ecr 843342387], length 192
02:48:33.962744 IP 10.1.22.2.22 > 192.168.3.100.61644: Flags [P.], seq
192:416, ack 1, win 49, options [nop,nop,TS val 1422334878 ecr 843342387],
length 224
```