**tcpdump -xx -XX - dump packet header and data in hex and ASCII format**

To troubleshoot network protocols or network filters (firewalls and Intrussion Detection Systems) sometimes it is required to look at the packets hex dump either live or at a packet capture taken while the investigated issue was happening. **tcpdump -xx** can be used to print packet hex data and layer 2 header information (when debugging ethernet mac address).

**Quote:**

-x When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The  
smaller of the entire packet or snaplen bytes will be printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Etherâ€  
net), the padding bytes will also be printed when the higher layer packet is shorter than the required padding.  
  
-xx When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, including its link level header, in hex.  
  
-X When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and  
ASCII. This is very handy for analysing new protocols.  
  
-XX When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, including its link level header, in hex and  
ASCII.

**Example on how to print packet data and ethernet header in hex**

**Code:**

$ tcpdump -nni vmx0 -e -xx -XX icmp and host 10.1.22.2  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on vmx0, link-type EN10MB (Ethernet), capture size 65535 bytes  
capability mode sandbox enabled  
10:22:52.750213 00:0c:29:2a:32:b0 > 80:71:1f:39:61:c8, ethertype IPv4 (0x0800), length 98: 10.1.1.54 > 10.1.22.2: ICMP echo request, id 20515, seq 38, length 64  
        0x0000:  8071 1f39 61c8 000c 292a 32b0 0800 4500  .q.9a...)\*2...E.  
        0x0010:  0054 cefe 0000 8001 4071 0a01 0136 0a01  .T......@q...6..  
        0x0020:  1602 0800 6002 5023 0026 550a 956c 000b  ....`.P#.&U..l..  
        0x0030:  722f 0809 0a0b 0c0d 0e0f 1011 1213 1415  r/..............  
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%  
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()\*+,-./012345  
        0x0060:  3637                                     67  
10:22:52.806495 80:71:1f:39:61:c8 > 00:0c:29:2a:32:b0, ethertype IPv4 (0x0800), length 98: 10.1.22.2 > 10.1.1.54: ICMP echo reply, id 20515, seq 38, length 64  
        0x0000:  000c 292a 32b0 8071 1f39 61c8 0800 4500  ..)\*2..q.9a...E.  
        0x0010:  0054 dfbd 0000 3e01 71b2 0a01 1602 0a01  .T....>.q.......  
        0x0020:  0136 0000 6802 5023 0026 550a 956c 000b  .6..h.P#.&U..l..  
        0x0030:  722f 0809 0a0b 0c0d 0e0f 1011 1213 1415  r/..............  
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%  
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()\*+,-./012345  
        0x0060:  3637                                     67

in above packets tcpdump prints the source and destination mac addresses involved in this transaction (00:0c:29:2a:32:b0 > 80:71:1f:39:61:c8) and they can also be seen in the hex dump, destination mac address at offset 0x0000 to offset 0x0005, followed by source mac starting at offset 0x0006 to 0x0011.  
  
Next, at offset 0x0012 it is the thernet type 0800 - IPv4 followed by offset 0x0013 ( hex 0x45, binary 01000101) representing IP version 4 (binary 0100) and IP header length 20 (0101 = 5 \* 4). The remaining fields of in this **tcpdump -xx -XX** output are: remaining ipv4 header fields, icmp header and ICMP data.