INR Lab Assignment:
Network Tools*

Layers 1 and 2 - Physical and Data Link

Task 1. Find out what network cards your server has. To what type of computer expansion bus are they connected? What is the speed of this interconnecting bus in mebibytes per second? Hint: lspci

Answer:

According to the source and your hint, to see which network car my server has:

```
kotaiba@bristol:~$ lspci | grep "Ethernet"
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM5716
Gigabit Ethernet (rev 20)
01:00.1 Ethernet controller: Broadcom Corporation NetXtreme II BCM5716
Gigabit Ethernet (rev 20)
```

what type of computer expansion bus are they connected?

We need to use lspci with -t “Show a tree-like diagram containing all buses”

```
kotaiba@bristol:~$ lspci -t
-[0000:00]-+-00.0
           +-1a.0
           +-1c.0-[01]--+-00.0
           |            \-00.1
           +-1d.0
           +-1e.0-[02]----03.0
           +-1f.0
           +-1f.2
           \-1f.3
```

As we see they are connected to 00:1c.0, so now we can see more information about that device:

```
kotaiba@bristol:~$ sudo lspci -vv -s 00:1c.0
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family
PCI Express Root Port 1 (rev b4) (prog-if 00 [Normal decode])
    Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR+ FastB2B- DisINTx+
    Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
    Latency: 0, Cache Line Size: 64 bytes
    Interrupt: pin A routed to IRQ 54
    Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
    Memory behind bridge: c0000000-c3ffffff
    Secondary status: 66MHz- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort+ <SERR- <PERR-
    BridgeCtl: Parity- SERR+ NoISA- VGA- MAbort- >Reset- FastB2B-
        PriDiscTmr- SecDiscTmr- DiscTmrStat- DiscTmrSERREn-
    Capabilities: [40] Express (v2) Root Port (Slot+), MSI 00
        DevCap:    MaxPayload 128 bytes, PhantFunc 0
            ExtTag- RBE+
        DevCtl:    Report errors: Correctable- Non-Fatal- Fatal+
Unsupported-
            RlxdOrd- ExtTag- PhantFunc- AuxPwr- NoSnoop-
            MaxPayload 128 bytes, MaxReadReq 128 bytes
        DevSta:    CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr+
TransPend-
        LnkCap:    Port #1, Speed 5GT/s, Width x4, ASPM L0s L1, Exit Latency
L0s <512ns, L1 <4us
            ClockPM- Surprise- LLActRep+ BwNot- ASPMOptComp-
        LnkCtl:    ASPM Disabled; RCB 64 bytes Disabled- CommClk+
            ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
        LnkSta:    Speed 2.5GT/s, Width x4, TrErr- Train- SlotClk+ DLActive+
BWMgmt+ ABWMgmt-
        SltCap:    AttnBtn- PwrCtrl- MRL- AttnInd- PwrInd- HotPlug-
Surprise-
            Slot #0, PowerLimit 25.000W; Interlock- NoCompl+
        SltCtl:    Enable: AttnBtn- PwrFlt- MRL- PresDet- CmdCplt- HPIrq-
LinkChg-
            Control: AttnInd Unknown, PwrInd Unknown, Power- Interlock-
        SltSta:    Status: AttnBtn- PowerFlt- MRL- CmdCplt- PresDet+
Interlock-
            Changed: MRL- PresDet- LinkState+
        RootCtl: ErrCorrectable- ErrNon-Fatal- ErrFatal+ PMEIntEna+
CRSVisible-
        RootCap: CRSVisible-
        RootSta: PME ReqID 0000, PMEStatus- PMEPending-
        DevCap2: Completion Timeout: Range BC, TimeoutDis+, LTR-, OBFF Not
Supported ARIFwd-
        DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF
Disabled ARIFwd-
        LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
             Transmit Margin: Normal Operating Range,
EnterModifiedCompliance- ComplianceSOS-
             Compliance De-emphasis: -6dB
        LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete-,
EqualizationPhase1-
             EqualizationPhase2-, EqualizationPhase3-,
LinkEqualizationRequest-
    Capabilities: [80] MSI: Enable+ Count=1/1 Maskable- 64bit-
        Address: fee00258  Data: 0000
    Capabilities: [90] Subsystem: Dell 6 Series/C200 Series Chipset Family
PCI Express Root Port 1
```

```
    Capabilities: [a0] Power Management version 2
        Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0+,D1-,D2-
,D3hot+,D3cold+)
        Status: D0 NoSoftRst- PME-Enable- DSel=0 DScale=0 PME-
    Kernel driver in use: pcieport
    Kernel modules: shpchp
```

As we see above the LnkSta (current status which is the actual device status), (i will escape LnkCap because it is the device capabilities not the current) is what we need, it's speed is Speed 2.5GT/s (2.5^9 transfers per second).

So my transfer rate: 2.5 GT/s and my width is = ×4 (which means according to source 3, my speed is 1GB/s which is 953.67 mebibytes " using converter", that the speed of the interconnecting bus.

Source:

1- https://help.ubuntu.com/stable/ubuntu-help/net-wireless-troubleshooting-hardware-check.html, 2- https://community.mellanox.com/docs/DOC-2496, 3- https://en.wikipedia.org/wiki/PCI_Express#History_and_revisions

Task 2. What is the current speed of the network interface? What offload features are enabled? Briefly explain the purpose of the tcp-segmentation-offload feature. Hint: ethtool

Answer: First, let's define ethtool: "is a utility for Linux kernel-based operating system for displaying and modifying some parameters of network interface controllers (NICs) and their device drivers."

Install it:

```
kotaiba@bristol:~$ sudo apt install ethtool
```

Now, in order to find the current speed of the network interface:

```
kotaiba@bristol:~$ ethtool eno1
Settings for eno1:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 1
    Transceiver: internal
    Auto-negotiation: on
    MDI-X: on
Cannot get wake-on-lan settings: Operation not permitted
    Link detected: yes
```

As we see above the Speed: 1000Mb/s .

In order to see what offload features are enabled, I will use the –show-offload option:

```
kotaiba@bristol:~$ ethtool --show-offload eno1
Features for eno1:
rx-checksumming: on
tx-checksumming: on
    tx-checksum-ipv4: on
    tx-checksum-ip-generic: off [fixed]
    tx-checksum-ipv6: on
    tx-checksum-fcoe-crc: off [fixed]
    tx-checksum-sctp: off [fixed]
scatter-gather: on
    tx-scatter-gather: on
    tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
    tx-tcp-segmentation: on
    tx-tcp-ecn-segmentation: on
    tx-tcp6-segmentation: on
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: on [fixed]
tx-vlan-offload: on
ntuple-filters: off [fixed]
receive-hashing: on
highdma: on [fixed]
rx-vlan-filter: off [fixed]
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-ipip-segmentation: off [fixed]
tx-sit-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
```

```
rx-fcs: off [fixed]
rx-all: off [fixed]
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off [fixed]
busy-poll: off [fixed]
hw-tc-offload: off [fixed]
```

As we got from above, the tcp segmentation offload:

```
tcp-segmentation-offload: on
    tx-tcp-segmentation: on
    tx-tcp-ecn-segmentation: on
    tx-tcp6-segmentation: on
```

TCP segmentation offload (TSO): The process of segmentation is “when a system needs to send large chunks of data out over a computer network, the chunks first need breaking down into smaller segments that can pass through all the network elements like routers and switches between the source and destination computers. Often the TCP protocol in the host computer performs this segmentation. Offloading this work to the NIC is called TCP segmentation offload (TSO).”

Sources:

1- https://en.wikipedia.org/wiki/Ethtool, 2-https://en.wikipedia.org/wiki/Large_send_offload

Task 3. What is the MAC address of the OS3 router facing your server? Can you infer the manufacturer of the network card? What about the MAC address of eth0/eno1 and its manufacturer? Hint: arp

Answer:

In order to see the MAC address of the OS3 router:

```
kotaiba@bristol:~$ arp -i eno1
Address                  HWtype  HWaddress           Flags Mask
Iface
wakefield.studlab.os3.n  ether   d4:ae:52:bf:e4:b3   C
eno1
router.studlab.os3.nl    ether   f8:b1:56:2f:b5:23   C
eno1
```

Now, I will check online for the vendor using website in source[1]:

It gave me f8:b1:56:2f:b5:23 –> Dell Inc.

Now, let's check the MAC address of eno1 of my server and its manufacturer:

```
kotaiba@bristol:~$ ifconfig | grep eno1
eno1      Link encap:Ethernet  HWaddr d4:ae:52:bf:e4:da
```

Again, using the same website, it gave me d4:ae:52:bf:e4:da -> Dell Inc.

Source:

1- https://www.macvendorlookup.com/

Task 4. Assuming that you have completed the previous lab, what interfaces are part of the xenbr0 bridge? What MAC addresses has this bridge learned so far? Hint: brctl

Answer:

I will use brctl “ethernet bridge administration” for this.

```
kotaiba@bristol:~$ brctl showmacs xenbr0
port no mac addr        is local?    ageing timer
  1 fe:ff:ff:ff:ff:ff    yes           0.00
  1 fe:ff:ff:ff:ff:ff    yes           0.00
```

The xenbr0 connected to the outside via the eno1 interface. Each guest virtually connects to the backend domain via a backend virtual network device, which is “vif1.0” in my case.

So now, let's checl its address:

```
kotaiba@bristol:~$ arp -i xenbr0
Address                  HWtype  HWaddress           Flags Mask
Iface
Guest-01                 ether   00:16:3e:8a:c4:65   C
xenbr0
```

Now, on my guest vm:

```
root@Guest-01:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:8a:c4:65
          inet addr:145.100.108.82  Bcast:145.100.108.95
Mask:255.255.255.240
          inet6 addr: fe80::216:3eff:fe8a:c465/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1447 errors:0 dropped:0 overruns:0 frame:0
          TX packets:522 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:157727 (157.7 KB)  TX bytes:87153 (87.1 KB)
```

As we see above, both MAC addresses are the same.

Task 5. How many bytes did your eth0/eno1 interface receive since boot? The kernel uses an unsigned long long variable for the RX bytes counter. How much traffic (in GB) must the server receive for this value to overflow? Hint: ifconfig

Answer:

The “RX byes” recived since boot is:

```
kotaiba@bristol:~$ ifconfig eno1 | grep "RX"
          RX packets:676133 errors:0 dropped:0 overruns:0 frame:0
          RX bytes:83551350 (83.5 MB)  TX bytes:73176880 (73.1 MB)
```

An unsigned long long integer, as in the C standard is 64 bit, has a size of 2^64. So the server will overflow at 1.8446744^19 bytes = 18446744073.709552765 GB

Source:

1-
https://stackoverflow.com/questions/5836329/how-many-bytes-is-unsigned-long-long/5836380#5836380

Task 6. What is the MTU setting for eth0/eno1? When do you think it should be increased? When do you think it should be decreased? Hint: ip link, ifconfig

Answer:

First, what is MTU “Maximum Transmission Unit, is the maximum size of a single data unit of digital communications that can be transmitted over a network.”

Get my eno1 MTU:

```
kotaiba@bristol:~$ ip link | grep "eno1"
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
```

The eno1 MTU is 1500 bytes. It decrease if the hardware does not allow for a large MTU. Large packets occupy a slow link for more time than a smaller packet, causing greater delays to subsequent packets, and increasing lag and minimum latency. If the hardware allows and the packet sizes is more than 1500 bytes it should be increased. A larger MTU also means processing of fewer packets for the same amount of data. In some systems, per-packet-processing can be a critical performance limitation. However, the Ethernet only support up to 1500 bytes per packet.

Source:

1- https://en.wikipedia.org/wiki/Maximum_transmission_unit, 2- https://www.lifewire.com/definition-of-mtu-817948

Layer 3 - Network

Task 7. What is the default gateway on your server? Why is there an explicit route to the OS3 network? If you would delete this latter route will you be able to send traffic to your default gateway? Why?

Answer:

The default gateway on my server:

```
kotaiba@bristol:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
0.0.0.0         145.100.104.161 0.0.0.0         UG    0      0        0 eno1
145.100.104.160 0.0.0.0         255.255.255.224 U     0      0        0 eno1
145.100.108.80  0.0.0.0         255.255.255.240 U     0      0        0
xenbr0
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0
xenbr1
```

As we see the gateway is 145.100.104.161 which is “router.studlab.”

there is an explicit route to the OS3 network because it is our link to the internet and it knows how to forward the packets to the outside world.

No I will not able to send traffic to my default gateway, since my server has no information of how to reach the default gateway if we delete it. and it will not know where the packet should go.

Received a feedback on this question **“Please reread the question.”**

Fix:

- My default gateway is:

```
root@bristol:/home/kotaiba# ip route | grep default
default via 145.100.104.161 dev eno1
```

- Why is there an explicit route to the OS3 network?

Because if it's not exist my server wouldn't know where traffic that doesn't match any other entry in the routing table should be sent to (in other word, “if you can't find it in the routing table, send it to the default gateway”).

- If you would delete this latter route will you be able to send traffic to your default gateway? Why?

Yes, I will still be able to sent traffic to IP address of gateway (145.100.104.161) because we can get this address again by using ARP since it is in the same broadcast domain. However, my server will not be able to reach addresses outside this broadcast domain.

Task 8. Perform a traceroute to bad.horse. Why does it stop after 30 hops? How can you increase this number? Provide the full traceroute output. Hint: mtr, traceroute

Answer:

```
kotaiba@bristol:~$ traceroute bad.horse
traceroute to bad.horse (162.252.205.157), 30 hops max, 60 byte packets
 1  * * *
 2  AE3.1664.JNR01.Asd002A.surf.net (145.145.19.190)  74.604 ms  74.580 ms
74.547 ms
 3  uk-hex.nordu.net (109.105.98.109)  7.052 ms  7.032 ms  7.000 ms
 4  nl-sar.nordu.net (109.105.97.124)  7.574 ms  7.560 ms  7.537 ms
 5  us-man.nordu.net (109.105.97.64)  92.699 ms  92.678 ms  92.638 ms
 6  * * *
 7  e6-1.cr1.lga12.amcbb.net (208.68.168.149)  95.486 ms  97.981 ms  95.469
ms
 8  e2-20.cr2.lga11.amcbb.net (69.9.32.221)  94.218 ms  102.622 ms  103.180
ms
 9  sandwichnet.dmarc.lga11.atlanticmetro.net (208.68.168.214)  92.916 ms
95.634 ms  93.050 ms
10  bad.horse (162.252.205.130)  98.302 ms  97.403 ms  96.801 ms
11  bad.horse (162.252.205.131)  98.221 ms  101.032 ms  103.418 ms
12  bad.horse (162.252.205.132)  111.868 ms  111.352 ms  108.546 ms
13  bad.horse (162.252.205.133)  108.118 ms  114.965 ms  118.581 ms
14  he.rides.across.the.nation (162.252.205.134)  122.635 ms  120.643 ms
128.738 ms
15  the.thoroughbred.of.sin (162.252.205.135)  122.354 ms  119.245 ms
119.201 ms
16  he.got.the.application (162.252.205.136)  128.589 ms  126.504 ms
131.290 ms
17  that.you.just.sent.in (162.252.205.137)  135.279 ms  135.218 ms  135.195
ms
18  it.needs.evaluation (162.252.205.138)  136.762 ms  139.868 ms  136.587
ms
19  so.let.the.games.begin (162.252.205.139)  141.386 ms  147.562 ms
148.436 ms
```

```
20  a.heinous.crime (162.252.205.140)  154.278 ms  149.708 ms  151.088 ms
21  a.show.of.force (162.252.205.141)  153.585 ms  153.180 ms  151.904 ms
22  a.murder.would.be.nice.of.course (162.252.205.142)  157.474 ms  152.659
ms  154.129 ms
23  bad.horse (162.252.205.143)  167.010 ms  164.327 ms  158.252 ms
24  bad.horse (162.252.205.144)  169.914 ms  167.395 ms  170.398 ms
25  bad.horse (162.252.205.145)  173.502 ms  170.807 ms  170.784 ms
26  he-s.bad (162.252.205.146)  181.645 ms  173.300 ms  181.937 ms
27  the.evil.league.of.evil (162.252.205.147)  189.685 ms  188.847 ms
188.772 ms
28  is.watching.so.beware (162.252.205.148)  185.853 ms  190.654 ms  192.725
ms
29  the.grade.that.you.receive (162.252.205.149)  197.871 ms  203.034 ms
205.343 ms
30  will.be.your.last.we.swear (162.252.205.150)  206.566 ms  208.934 ms
208.045 ms
```

As we see in the above output “traceroute to bad.horse (162.252.205.157), 30 hops max, 60 byte packets”, it stops after 30 hops because that is the default number of hops for tracerout command.

In order to increase it, I checked the tracerout man page “-m max_ttl Specifies the maximum number of hops (max time-to-live value) traceroute will probe. The default is 30.”

Let's test it for 31 hops:

```
kotaiba@bristol:~$ traceroute bad.horse -m 31
traceroute to bad.horse (162.252.205.157), 31 hops max, 60 byte packets
 1  * * router.studlab.os3.nl (145.100.104.161)  0.494 ms
 2  AE3.1664.JNR01.Asd002A.surf.net (145.145.19.190)  0.443 ms  0.422 ms
0.381 ms
 3  uk-hex.nordu.net (109.105.98.109)  7.016 ms  7.000 ms  6.962 ms
 4  nl-sar.nordu.net (109.105.97.124)  7.501 ms  7.488 ms  7.528 ms
 5  us-man.nordu.net (109.105.97.64)  92.620 ms us-man.nordu.net
(109.105.97.139)  95.297 ms us-man.nordu.net (109.105.97.64)  92.645 ms
 6  * * *
 7  e6-1.cr1.lga12.amcbb.net (208.68.168.149)  95.937 ms  95.915 ms  95.426
ms
 8  e2-20.cr2.lga11.amcbb.net (69.9.32.221)  95.507 ms  95.490 ms  92.988 ms
 9  sandwichnet.dmarc.lga11.atlanticmetro.net (208.68.168.214)  94.576 ms
99.169 ms  93.962 ms
10  bad.horse (162.252.205.130)  99.064 ms  96.646 ms  99.899 ms
11  bad.horse (162.252.205.131)  106.328 ms  102.450 ms  99.038 ms
12  bad.horse (162.252.205.132)  106.793 ms  106.656 ms  106.616 ms
13  bad.horse (162.252.205.133)  110.707 ms  116.120 ms  111.463 ms
14  he.rides.across.the.nation (162.252.205.134)  120.608 ms  121.650 ms
124.120 ms
15  the.thoroughbred.of.sin (162.252.205.135)  123.541 ms  124.459 ms
118.316 ms
16  he.got.the.application (162.252.205.136)  124.273 ms  124.407 ms
128.036 ms
17  that.you.just.sent.in (162.252.205.137)  130.766 ms  134.919 ms  130.950
```

```
ms
18  it.needs.evaluation (162.252.205.138)  138.186 ms  141.182 ms  139.143
ms
19  so.let.the.games.begin (162.252.205.139)  143.407 ms  143.337 ms
141.022 ms
20  a.heinous.crime (162.252.205.140)  147.993 ms  150.129 ms  151.994 ms
21  a.show.of.force (162.252.205.141)  151.060 ms  145.727 ms  150.916 ms
22  a.murder.would.be.nice.of.course (162.252.205.142)  157.809 ms  155.342
ms  155.967 ms
23  bad.horse (162.252.205.143)  161.282 ms  164.287 ms  163.918 ms
24  bad.horse (162.252.205.144)  173.652 ms  163.874 ms  172.072 ms
25  bad.horse (162.252.205.145)  176.694 ms  173.445 ms  172.778 ms
26  he-s.bad (162.252.205.146)  178.899 ms  175.302 ms  177.332 ms
27  the.evil.league.of.evil (162.252.205.147)  183.070 ms  186.553 ms
184.633 ms
28  is.watching.so.beware (162.252.205.148)  188.438 ms  188.717 ms  184.391
ms
29  the.grade.that.you.receive (162.252.205.149)  195.430 ms  190.922 ms
193.730 ms
30  will.be.your.last.we.swear (162.252.205.150)  194.997 ms  194.840 ms
199.467 ms
31  so.make.the.bad.horse.gleeful (162.252.205.151)  202.073 ms  200.792 ms
200.993 ms
```

As we see, it works.

Task 9. What are the three built-in chains in the netfilter 'filter' table? Briefly explain what is the purpose of each chain.

Answer:

Filter is default table for iptables. So, if you don't define you own table, you'll be using filter table. Iptables's filter table has the following built-in chains.

- INPUT chain – Incoming to firewall. For packets coming to the local server.
- OUTPUT chain – Outgoing from firewall. For packets generated locally and going out of the local server.
- FORWARD chain – Packet for another NIC on the local server. For packets routed through the local server.

Source:

1- http://www.thegeekstuff.com/2011/01/iptables-fundamentals

Layer 4 - Transport

Task 10. What ports are currently open on your machine? What services do they belong to? Hint: netstat

Answer:

The opened ports on my machine:

```
kotaiba@bristol:~$ sudo netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       
PID/Program name
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      
4168/smbd
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN      
-
tcp        0      0 0.0.0.0:39337           0.0.0.0:*               LISTEN      
-
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      
4168/smbd
tcp        0      0 0.0.0.0:33868           0.0.0.0:*               LISTEN      
829/rpc.mountd
tcp        0      0 0.0.0.0:38348           0.0.0.0:*               LISTEN      
829/rpc.mountd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      
827/rpcbind
tcp        0      0 0.0.0.0:54224           0.0.0.0:*               LISTEN      
829/rpc.mountd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      
820/sshd
tcp        0      0 0.0.0.0:38555           0.0.0.0:*               LISTEN      
-
tcp6       0      0 :::445                  :::*                    LISTEN      
4168/smbd
tcp6       0      0 :::47357                :::*                    LISTEN      
829/rpc.mountd
tcp6       0      0 :::34208                :::*                    LISTEN      
829/rpc.mountd
tcp6       0      0 :::34529                :::*                    LISTEN      
-
tcp6       0      0 :::2049                 :::*                    LISTEN      
-
tcp6       0      0 :::139                  :::*                    LISTEN      
4168/smbd
tcp6       0      0 :::111                  :::*                    LISTEN      
827/rpcbind
```

```
tcp6       0      0 :::37108                :::*                    LISTEN
-
tcp6       0      0 :::22                   :::*                    LISTEN
820/sshd
tcp6       0      0 :::51416                :::*                    LISTEN
829/rpc.mountd
udp        0      0 0.0.0.0:54116           0.0.0.0:*
829/rpc.mountd
udp        0      0 0.0.0.0:37837           0.0.0.0:*
-
udp        0      0 0.0.0.0:1003            0.0.0.0:*
827/rpcbind
udp        0      0 0.0.0.0:50603           0.0.0.0:*
829/rpc.mountd
udp        0      0 0.0.0.0:2049            0.0.0.0:*
-
udp        0      0 0.0.0.0:68              0.0.0.0:*
645/dhclient
udp        0      0 0.0.0.0:111             0.0.0.0:*
827/rpcbind
udp        0      0 145.100.104.191:137     0.0.0.0:*
3978/nmbd
udp        0      0 145.100.104.163:137     0.0.0.0:*
3978/nmbd
udp        0      0 145.100.108.95:137      0.0.0.0:*
3978/nmbd
udp        0      0 145.100.108.81:137      0.0.0.0:*
3978/nmbd
udp        0      0 192.168.255.255:137     0.0.0.0:*
3978/nmbd
udp        0      0 192.168.0.11:137        0.0.0.0:*
3978/nmbd
udp        0      0 0.0.0.0:137             0.0.0.0:*
3978/nmbd
udp        0      0 145.100.104.191:138     0.0.0.0:*
3978/nmbd
udp        0      0 145.100.104.163:138     0.0.0.0:*
3978/nmbd
udp        0      0 145.100.108.95:138      0.0.0.0:*
3978/nmbd
udp        0      0 145.100.108.81:138      0.0.0.0:*
3978/nmbd
udp        0      0 192.168.255.255:138     0.0.0.0:*
3978/nmbd
udp        0      0 192.168.0.11:138        0.0.0.0:*
3978/nmbd
udp        0      0 0.0.0.0:138             0.0.0.0:*
3978/nmbd
udp        0      0 0.0.0.0:49666           0.0.0.0:*
829/rpc.mountd
udp6       0      0 :::1003                 :::*
```

```
827/rpcbind
udp6       0      0 :::46929                :::*
829/rpc.mountd
udp6       0      0 :::2049                 :::*
-
udp6       0      0 :::60076                :::*
-
udp6       0      0 :::111                  :::*
827/rpcbind
udp6       0      0 :::53372                :::*
829/rpc.mountd
udp6       0      0 :::53626                :::*
829/rpc.mountd
```

In general, rpc.mountd (used for NFS server deamon storage share, since I was working on the LS lab), sshd (for my SSH connection), smbd (also for SMB server deamon for filesharing “LS assignment”), dhclient is the Dynamic Host Configuration Protocol (DHCP) Client one would use to allow a client to connect to a DHCP server and so on.

In order to see each port with its corresponding service please go to “https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Security_Guide/ch-ports.html” you will get a list of ports with its services.

Task 11. How many unix sockets are currently created on your server? What are unix sockets used for? Hint: lsof

Answer:

A Unix domain socket or IPC socket (inter-process communication socket) is a data communications endpoint for exchanging data between processes executing on the same host operating system. Sockets are commonly used for client and server interaction. Typical system configuration places the server on one machine, with the clients on other machines. The clients connect to the server, exchange information, and then disconnect.

In order to list all Unix socket:

-U = means Unix Socket

```
kotaiba@bristol:~$ sudo lsof -U
COMMAND    PID             USER   FD   TYPE            DEVICE SIZE/OFF
NODE NAME
systemd      1             root   14u  unix 0xffff880227acf400      0t0
8985 /run/systemd/private type=STREAM
systemd      1             root   19u  unix 0xffff880089fa0400      0t0
46871 type=DGRAM
systemd      1             root   24u  unix 0xffff880227acec00      0t0
8984 /run/systemd/notify type=DGRAM
systemd      1             root   25u  unix 0xffff88022987c800      0t0
```

```
8991 /run/udev/control type=SEQPACKET
systemd      1              root   28u  unix 0xffff880227acf000      0t0
8989 /run/systemd/journal/dev-log type=DGRAM
systemd      1              root   29u  unix 0xffff880226aea000      0t0
11405 /run/lvm/lvmpolld.socket type=STREAM
systemd      1              root   31u  unix 0xffff8802274c5400      0t0
9088 /run/systemd/journal/stdout type=STREAM
systemd      1              root   32u  unix 0xffff8802291ebc00      0t0
11632 /run/systemd/journal/stdout type=STREAM
systemd      1              root   34u  unix 0xffff8802275abc00      0t0
12522 /run/systemd/journal/stdout type=STREAM
systemd      1              root   35u  unix 0xffff8802274a0800      0t0
12523 /run/systemd/journal/stdout type=STREAM
systemd      1              root   37u  unix 0xffff88022b7fa800      0t0
12565 /run/systemd/journal/stdout type=STREAM
systemd      1              root   38u  unix 0xffff8800bd074400      0t0
13777 /run/systemd/journal/stdout type=STREAM
systemd      1              root   39u  unix 0xffff8802299a6800      0t0
12627 /run/systemd/journal/stdout type=STREAM
systemd      1              root   40u  unix 0xffff88000560cc00      0t0
14012 /run/systemd/journal/stdout type=STREAM
systemd      1              root   41u  unix 0xffff8802291bfc00      0t0
14024 /run/systemd/journal/stdout type=STREAM
systemd      1              root   42u  unix 0xffff88022b7fb800      0t0
11358 /run/lvm/lvmetad.socket type=STREAM
systemd      1              root   43u  unix 0xffff88022b722000      0t0
12453 /run/rpcbind.sock type=STREAM
systemd      1              root   46u  unix 0xffff88022987d000      0t0
8993 /run/systemd/journal/syslog type=DGRAM
systemd      1              root   47u  unix 0xffff88022987cc00      0t0
8992 /run/systemd/fsck.progress type=STREAM
systemd      1              root   49u  unix 0xffff880226a7c400      0t0
12028 type=STREAM
systemd      1              root   52u  unix 0xffff88000560d800      0t0
13630 /var/run/dbus/system_bus_socket type=STREAM
systemd      1              root   53u  unix 0xffff88022987d400      0t0
8994 /run/systemd/journal/stdout type=STREAM
systemd      1              root   54u  unix 0xffff88022987d800      0t0
8995 /run/systemd/journal/socket type=DGRAM
systemd-j  244              root    4u  unix 0xffff88022987d400      0t0
8994 /run/systemd/journal/stdout type=STREAM
systemd-j  244              root    5u  unix 0xffff88022987d800      0t0
8995 /run/systemd/journal/socket type=DGRAM
systemd-j  244              root    6u  unix 0xffff880227acf000      0t0
8989 /run/systemd/journal/dev-log type=DGRAM
systemd-j  244              root   14u  unix 0xffff880000568fc00     0t0
9064 type=DGRAM
systemd-j  244              root   17u  unix 0xffff88000560cc00      0t0
14012 /run/systemd/journal/stdout type=STREAM
systemd-j  244              root   18u  unix 0xffff8802299a6800      0t0
12627 /run/systemd/journal/stdout type=STREAM
```

```
systemd-j  244             root   19u  unix 0xffff8802274c5400      0t0
9088 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   20u  unix 0xffff8802291ebc00      0t0
11632 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   22u  unix 0xffff8802291bfc00      0t0
14024 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   24u  unix 0xffff8802275abc00      0t0
12522 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   25u  unix 0xffff8802274a0800      0t0
12523 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   26u  unix 0xffff88022b7fa800      0t0
12565 /run/systemd/journal/stdout type=STREAM
systemd-j  244             root   27u  unix 0xffff8800bd074400      0t0
13777 /run/systemd/journal/stdout type=STREAM
lvmetad    253             root    1u  unix 0xffff8802274c5800      0t0
10470 type=STREAM
lvmetad    253             root    2u  unix 0xffff8802274c5800      0t0
10470 type=STREAM
lvmetad    253             root    3u  unix 0xffff88022b7fb800      0t0
11358 /run/lvm/lvmetad.socket type=STREAM
systemd-u  316             root    1u  unix 0xffff8802291e8400      0t0
11630 type=STREAM
systemd-u  316             root    2u  unix 0xffff8802291e8400      0t0
11630 type=STREAM
systemd-u  316             root    3u  unix 0xffff88022987c800      0t0
8991 /run/udev/control type=SEQPACKET
systemd-u  316             root    5u  unix 0xffff88022b769400      0t0
9177 type=DGRAM
systemd-u  316             root    7u  unix 0xffff88022b76ac00      0t0
9181 type=DGRAM
systemd-u  316             root    8u  unix 0xffff88022b768800      0t0
9182 type=DGRAM
rpc.idmap  478             root    5u  unix 0xffff88000560e000      0t0
13615 type=STREAM
rpc.idmap  478             root    6u  unix 0xffff88000560e400      0t0
13616 type=STREAM
accounts-  509             root    1u  unix 0xffff8802275ab400      0t0
13749 type=STREAM
accounts-  509             root    2u  unix 0xffff8802275ab400      0t0
13749 type=STREAM
accounts-  509             root    5u  unix 0xffff8802275bb400      0t0
12040 type=STREAM
rsyslogd   512           syslog    3u  unix 0xffff88022987d000      0t0
8993 /run/systemd/journal/syslog type=DGRAM
dbus-daem  516       messagebus    1u  unix 0xffff8802274a2000      0t0
12025 type=STREAM
dbus-daem  516       messagebus    2u  unix 0xffff8802274a2000      0t0
12025 type=STREAM
dbus-daem  516       messagebus    3u  unix 0xffff88000560d800      0t0
13630 /var/run/dbus/system_bus_socket type=STREAM
dbus-daem  516       messagebus    5u  unix 0xffff8802274a2c00      0t0
```

```
12552 type=DGRAM
dbus-daem  516       messagebus    8u  unix 0xffff880226aebc00      0t0
12555 type=STREAM
dbus-daem  516       messagebus    9u  unix 0xffff880226ae9800      0t0
12556 type=STREAM
dbus-daem  516       messagebus   10u  unix 0xffff880226a7d000      0t0
12557 /var/run/dbus/system_bus_socket type=STREAM
dbus-daem  516       messagebus   11u  unix 0xffff8802275b9000      0t0
12558 /var/run/dbus/system_bus_socket type=STREAM
dbus-daem  516       messagebus   12u  unix 0xffff880226ae9000      0t0
12590 /var/run/dbus/system_bus_socket type=STREAM
systemd-l  563             root    1u  unix 0xffff8802275ba000      0t0
12096 type=STREAM
systemd-l  563             root    2u  unix 0xffff8802275ba000      0t0
12096 type=STREAM
systemd-l  563             root    3u  unix 0xffff8802291be000      0t0
13778 type=DGRAM
systemd-l  563             root   12u  unix 0xffff880226ae8400      0t0
12589 type=STREAM
cron       570             root    1u  unix 0xffff8800bd077800      0t0
12155 type=STREAM
cron       570             root    2u  unix 0xffff8800bd077800      0t0
12155 type=STREAM
systemd-t  642 systemd-timesync    1u  unix 0xffff8802299a5000      0t0
9945 type=STREAM
systemd-t  642 systemd-timesync    2u  unix 0xffff8802299a5000      0t0
9945 type=STREAM
systemd-t  642 systemd-timesync    3u  unix 0xffff8802299a4800      0t0
9949 type=DGRAM
systemd-t  642 systemd-timesync    7u  unix 0xffff8802299a7400      0t0
9953 type=DGRAM
systemd-t  642 systemd-timesync    8u  unix 0xffff8802299a4000      0t0
9954 type=DGRAM
systemd-t  642 systemd-timesync    9u  unix 0xffff8802299a7c00      0t0
9955 type=DGRAM
systemd-t  642 systemd-timesync   10u  unix 0xffff8802299a6400      0t0
9956 type=DGRAM
dhclient   645             root    3u  unix 0xffff8802275bac00      0t0
12352 type=DGRAM
sshd       820             root    1u  unix 0xffff88000560c400      0t0
10075 type=STREAM
sshd       820             root    2u  unix 0xffff88000560c400      0t0
10075 type=STREAM
rpcbind    827             root    1u  unix 0xffff8802291bd800      0t0
10192 type=STREAM
rpcbind    827             root    2u  unix 0xffff8802291bd800      0t0
10192 type=STREAM
rpcbind    827             root    3u  unix 0xffff88022b722000      0t0
12453 /run/rpcbind.sock type=STREAM
rpc.mount  829             root   20u  unix 0xffff88022b722c00      0t0
10231 type=DGRAM
```

```
xenstored  903             root    3u  unix 0xffff8802299a5800      0t0
14129 /var/run/xenstored/socket type=STREAM
xenstored  903             root    4u  unix 0xffff8802299a7800      0t0
14130 /var/run/xenstored/socket_ro type=STREAM
xenstored  903             root    7u  unix 0xffff8802299a6000      0t0
14132 type=DGRAM
xenstored  903             root    8u  unix 0xffff8802275a8400      0t0
14136 /var/run/xenstored/socket type=STREAM
xenstored  903             root   12u  unix 0xffff8802275b8000      0t0
15672 /var/run/xenstored/socket type=STREAM
xenstored  903             root   13u  unix 0xffff8802043b0c00      0t0
14656 /var/run/xenstored/socket type=STREAM
xenstored  903             root   14u  unix 0xffff8800baeb3000      0t0
14156 /var/run/xenstored/socket type=STREAM
xenconsol  926             root    3u  unix 0xffff880227606800      0t0
15670 type=STREAM
qemu-syst  932             root   16u  unix 0xffff8802275ba400      0t0
12909 type=STREAM
xl        1118             root    6u  unix 0xffff880227428000      0t0
12939 type=STREAM
xl        1118             root   14u  unix 0xffff8802043b2000      0t0
14222 type=STREAM
nmbd      3978             root    6u  unix 0xffff880065fed800      0t0
48438 /var/lib/samba/private/msg.sock/3978 type=DGRAM
nmbd      3978             root   30u  unix 0xffff880065fee000      0t0
48456 /var/run/samba/nmbd/unexpected type=STREAM
smbd      4168             root    7u  unix 0xffff8801d1e8ec00      0t0
51494 /var/lib/samba/private/msg.sock/4168 type=DGRAM
smbd      4169             root    7u  unix 0xffff880089865400      0t0
52350 /var/lib/samba/private/msg.sock/4169 type=DGRAM
smbd      4171             root    7u  unix 0xffff880089864400      0t0
52351 /var/lib/samba/private/msg.sock/4171 type=DGRAM
smbd      4292             root    7u  unix 0xffff8800a532bc00      0t0
51958 /var/lib/samba/private/msg.sock/4292 type=DGRAM
smbd      4292             root   34u  unix 0xffff8800a5329c00      0t0
51967 type=DGRAM
sshd      4612             root    4u  unix 0xffff8800a53f8000      0t0
919422 type=DGRAM
sshd      4612             root    6u  unix 0xffff8800a53fa000      0t0
919426 type=STREAM
sshd      4621          kotaiba    4u  unix 0xffff8800a53f8000      0t0
919422 type=DGRAM
sshd      4621          kotaiba    5u  unix 0xffff8800a53f8c00      0t0
919425 type=STREAM
sshd      5567             root    6u  unix 0xffff88008b474000      0t0
943675 type=STREAM
sshd      5568             sshd    4u  unix 0xffff88008b477c00      0t0
943674 type=STREAM
sudo      5605             root    3u  unix 0xffff88008b475400      0t0
943851 type=STREAM
sudo      5605             root    8u  unix 0xffff88008b475000      0t0
```

```
943854 type=DGRAM
```

Now, to count the number of sockets we count the lines -1 (because of the first line information):

```
kotaiba@bristol:~$ sudo lsof -U | wc -l
107
```

107 - 1 = 106 sockets.

Source:

1- https://en.wikipedia.org/wiki/Unix_domain_socket, 2-
https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rzab6/howdosockets.htm

Layer 7 - Application

Task 12. How can you test that a machine is listening on a specific TCP port? Can you do the same for UDP? Why? Hint: nc, telnet

Answer:

In order to test we can use “Netcat: (often abbreviated to nc) is a computer networking utility for reading from and writing to network connections using TCP or UDP.”

-z' Specifies that nc should just scan for listening daemons, without sending any data to them. It is an error to use this option in conjunction with the -l option.

-v verbose

Test os3.nl if its listening on TCP port 443:

```
kotaiba@bristol:~$ nc -zvvv www.os3.nl 443
Connection to www.os3.nl 443 port [tcp/https] succeeded!
```

As we know that UDP is connectionless, which means we don't get response or anything that a specific UDP port is listening or not, However netcat also support UPD ports check.

-u' Use UDP instead of the default option of TCP.

```
kotaiba@bristol:~$ nc -v -u www.google.nl 443
Connection to www.google.nl 443 port [udp/https] succeeded!
```

The problem here, is that netcat always shows that UDP ports are open. Most scanners assume that if there is no response that means the port is open, and if there is an ICMP error message that means the port is closed. There is no definite way to determine which port is closed or open in case of UDP ports since it is connectionless.

Source:

1- https://linux.die.net/man/1/nc, 2-
https://stackpointer.io/unix/unix-linux-netcat-check-port-open/511/

Task 13. What is the type and version of the webserver that serves www.os3.nl? Hint: curl, wget

Answer:

Type and version of www.os3.nl, I will use curl with -I:

```
-i, --include
Include the HTTP response headers in the output. The HTTP response headers
can include things like server name, cookies, date of the document, HTTP
version and more...
```

Test it:

```
kotaiba@bristol:~$ curl -I www.os3.nl
HTTP/1.1 302 Found
Date: Sun, 05 Nov 2017 01:55:39 GMT
Server: Apache/2.4.10 (Debian)
Strict-Transport-Security: max-age=31536000
Location: https://www.os3.nl/
Content-Type: text/html; charset=iso-8859-1
```

As we see, type “Apache” and version “2.4.10 (Debian)”.

Packet capturing

Task 14. Make sure that the Guest-01 VM is turned off. On your Dom0 start tcpdump and make sure it listens only on the xenbr0 interface. All captured packets should be saved to a file called capture.pcap. Start Guest-01 and after it boots ping www.os3.nl with a packet count of 10, each packet having a payload of 1024 bytes. Once the ping process exits, stop tcpdump and inspect the captured file using tcpdump or Wireshark. What is the size of each ICMP packet? Why is it not 1024?

Answer:

First, let's shutdown the Guest-01:

```
kotaiba@bristol:~$ sudo xl list
[sudo] password for kotaiba:
Name                                        ID   Mem VCPUs  State   Time(s)
Domain-0                                     0  5989     4     r-----
2527.7
Guest-01                                     1  1024     2     -b----
236.6
kotaiba@bristol:~$ sudo xl shutdown 1
Shutting down domain 1
```

Now, Let's start tcpdump and make sure it listens only on the xenbr0 interface and save the output to capture.pcap:

```
kotaiba@bristol:~$ sudo tcpdump -ni xenbr0 -s0 -w capture.pca
tcpdump: listening on xenbr0, link-type EN10MB (Ethernet), capture size
262144 bytes
```

Since I have a image for my Guest-01, I will restore it to perform the ping and login:

```
kotaiba@bristol:~$ sudo xl restor cold_bristol
Loading new save file cold_bristol (new xl fmt info 0x3/0x0/1506)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0x1aa838, dom 0, evt 1
xc: info: Console: mfn 0x1aa837, dom 0, evt 2

kotaiba@bristol:~$ sudo xl consol Guest-01
```

Now, we need to ping www.os3.nl with a packet count of 10, each packet having a payload of 1024 bytes:

```
root@Guest-01:~# ping -c 10 -s 1024 145.100.96.70
PING 145.100.96.70 (145.100.96.70) 1024(1052) bytes of data.

--- 145.100.96.70 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 8999ms
```

Now, let's see the captured file:

```
kotaiba@bristol:~$ tcpdump -tttt -r capture.pca
reading from file capture.pca, link-type EN10MB (Ethernet)
2017-11-05 03:12:59.812815 ARP, Request who-has Guest-01 tell Guest-01,
length 28
2017-11-05 03:12:59.812829 IP6 fe80::216:3eff:fe8a:c465 > ip6-allnodes:
ICMP6, neighbor advertisement, tgt is fe80::216:3eff:fe8a:c465, length 32
```

```
2017-11-05 03:12:59.814705 IP6 fe80::9480:eff:fe25:f7e > ff02::16: HBH
ICMP6, multicast listener report v2, 2 group record(s), length 48
2017-11-05 03:12:59.815981 IP6 fe80::216:3eff:fe8a:c465 > ff02::16: HBH
ICMP6, multicast listener report v2, 1 group record(s), length 28
2017-11-05 03:13:00.694727 IP6 fe80::9480:eff:fe25:f7e > ff02::16: HBH
ICMP6, multicast listener report v2, 2 group record(s), length 48
2017-11-05 03:13:00.720103 IP6 fe80::216:3eff:fe8a:c465 > ff02::16: HBH
ICMP6, multicast listener report v2, 1 group record(s), length 28
2017-11-05 03:15:40.516606 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 1, length 1032
2017-11-05 03:15:41.516363 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 2, length 1032
2017-11-05 03:15:42.516419 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 3, length 1032
2017-11-05 03:15:43.516463 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 4, length 1032
2017-11-05 03:15:44.516509 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 5, length 1032
2017-11-05 03:15:45.516554 IP Guest-01 > www.os3.nl: ICMP echo request, id
13335, seq 6, length 1032
```

As we notice above, The ICMP packets size is 1032 bytes. This is because the ICMP header has a size of 8 bytes. which means 1024 (the original size of 1024 bytes ping)+ 8 (ICMP header) = 1032 bytes.

Source:

1- https://osqa-ask.wireshark.org/questions/19054/tcpdump-text-output-to-pcap, 2- http://www.thegeekstuff.com/2010/08/tcpdump-command-examples