Notes on RHCE Exam Preparation

Isaac Hailperin `<isaac.hailperin@gmail.com>`

Revision History

Revision 0.1	2015-03	IH

Table of Contents

1. Preface

These are notes I took during preparation for the RHCE exam. I followed the RHCE book by Markus Frei, and also read some of the official Redhat documentation. I have omited topics that I felt comfortable with already, so this is by no means a complete study guide.

To actually try out things I had two virtual boxes with centos 7.0 installed.

2. Teaming

A good intrucution to managing teams with nmcli is given by

- RedHat [https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Configure_a_Network_Team_Using-the_Command_Line.html] and
- Fedora [http://docs.fedoraproject.org/en-US/Fedora/20/html/Networking_Guide/sec-Configure_Network_Teaming_Using_nmcli.html]

2.1. Teaming with nmcli

Create a new team connection using roundrobin with the two interfaces enp0s9 and enp0s10:

```
nmcli connection add type team con-name team0  ifname veteam0
#nmcli connection modify team0 ipv4.addresses  "10.23.23.77/24 10.23.23.1 "
nmcli connection modify team0 team.config roundrobin.conf
[root@rhce1 ~]# cat roundrobin.conf
{
        "device":               "team0",
        "runner":               {"name": "roundrobin"},
        "ports":                {"enp0s9": {}, "enp0s10": {}}
}
# add slave interfaces
nmcli connection add type team-slave con-name team0-port0 ifname enp0s9 master team0
nmcli connection add type team-slave con-name team0-port1 ifname enp0s10 master team0
```

Each slave interface needs to be configured with a seperate connection of type "team-slave".

The above works, but bringing team0 down and up again without a reboot does not work. For some reason, when bringng team0 down, the devices it uses are disconnected. If you try to reconnect them first

```
[root@rhce1 ~]# nmcli con down team0
[root@rhce1 ~]# nmcli dev st
DEVICE   TYPE      STATE         CONNECTION
enp0s3   ethernet  connected     Wired connection 1
enp0s8   ethernet  connected     hostonly
enp0s10  ethernet  disconnected  --
enp0s9   ethernet  disconnected  --
lo       loopback  unmanaged     --
[root@rhce1 ~]# nmcli dev connect enp0s9
Error: Device activation failed: The device has no connections available.
[root@rhce1 ~]# nmcli dev st
DEVICE   TYPE      STATE         CONNECTION
enp0s3   ethernet  connected     Wired connection 1
enp0s8   ethernet  connected     hostonly
enp0s10  ethernet  disconnected  --
enp0s9   ethernet  unavailable   --
lo       loopback  unmanaged     --
[root@rhce1 ~]# nmcli dev connect enp0s10
Device 'enp0s10' successfully activated with '1ad12cb7-767f-4886-b222-692560498b0d'.

[root@rhce1 ~]# nmcli con up team0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/Ac
[root@rhce1 ~]# nmcli dev st
DEVICE   TYPE      STATE        CONNECTION
enp0s10  ethernet  connected    team0-port1
enp0s3   ethernet  connected    Wired connection 1
enp0s8   ethernet  connected    hostonly
veteam0  team      connected    team0
```

```
enp0s9   ethernet  unavailable  --
lo       loopback  unmanaged    --
```

You can activate team0 again, but with just one interface.

2.2. Teaming with traditional ifcfg

The following traditional configuration works:

```
[root@rhce1 network-scripts]# for i in ifcfg-enp0s* ifcfg-team0 ;do echo $i;echo =======
ifcfg-enp0s10
===========
DEVICE="enp0s10"
ONBOOT="yes"
DEVICETYPE="TeamPort"
TEAM_MASTER="team0"

ifcfg-enp0s9
===========
DEVICE="enp0s9"
ONBOOT="yes"
DEVICETYPE="TeamPort"
TEAM_MASTER="team0"

ifcfg-team0
===========
TEAM_MASTER="team0"
DEVICE="team0"
DEVICETYPE="Team"
ONBOOT="yes"
BOOTPROTO=none
TEAM_CONFIG='{"runner": {"name": "roundrobin"}}'
IPADDR=10.23.23.77
GATEWAY=10.23.23.1
```

To active this, restart the networking service:

```
systemctl restart network.service
```

3. IPv6

You can activate ipv6 privacy extensions:

```
# vim /etc/sysctl.conf
# sysctl -p
net.ipv6.conf.enp0s3.use_tempaddr = 2
```

This will cause the autogenerated ipv6 address to be randomized and renewed every 24 hours.

You can disable ipv6 like this:

```
# vim /etc/sysctl.conf
# sysctl -p
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
```

Question: what is the difference in /proc/sys/net/ipvX/conf/ between "all" and "default"?

4. Routing

4.1. Configure static routes with iproute - nonpersistent

Note: all route table manipulations with ip are lost after reboot.

Show routes:

```
ip route
```

Delete a route

```
ip route del <route>
```

Example:

```
[root@rhce1 ~]# ip r
default via 10.23.23.99 dev enp0s8  proto static  metric 1024
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
10.23.23.0/24 dev enp0s8  proto kernel  scope link  src 10.23.23.51
10.23.23.0/24 dev veteam0  proto kernel  scope link  src 10.23.23.104
[root@rhce1 ~]# ip r del 10.23.23.0/24 dev veteam0
[root@rhce1 ~]# ip r
default via 10.23.23.99 dev enp0s8  proto static  metric 1024
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
10.23.23.0/24 dev enp0s8  proto kernel  scope link  src 10.23.23.51
```

You can add static route using following command:

```
ip route add {NETWORK} via {IP} dev {DEVICE}
```

Add a new default route:

```
ip r add default via <default gw>
```

Example:

```
[root@rhce1 ~]# ip r add default via 10.0.2.2
[root@rhce1 ~]# ip r
default via 10.0.2.2 dev enp0s3
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
10.23.23.0/24 dev enp0s8  proto kernel  scope link  src 10.23.23.51
10.23.23.0/24 dev veteam0  proto kernel  scope link  src 10.23.23.104
```

In order to make this persistent, you would need to put this in the appropriate file in

```
/etc/sysconfig/network-scripts/route-<interface>
```

E.g.

```
GATEWAY0=192.168.1.254
NETMASK0=255.255.255.0
ADDRESS0=192.168.55.0
GATEWAY1=10.164.234.112
```

```
NETMASK1= 255.255.255.240
ADDRESS1=10.164.234.132
```

The default gateway goes into

```
/etc/sysconfig/network # parameter is called GATEWAY
```

4.2. Configuring static routes with nmcli - persistent

To configure static routes using the nmcli tool, the interactive editor mode must be used:

```
nmcli con edit <con name>
```

```
nmcli> set ipv4.routes 192.168.122.0/24 10.10.10.1
nmcli>
nmcli> save persistent
```

To set the default route with NetworkManager, make sure only that interface providing the default gateway can set this (from https://mail.gnome.org/archives/networkmanager-list/2014-July/msg00080.html):

NM automatically sets the default route based on two things:

1. interface priority - all interfaces have a priority and if two interfaces are active, and **not** prevented from getting the default route (see #2), the one with the highest priority wins. Right now, that’s a static ordering but we’re exploring how to make that dynamic.

2. the "never-default" option: you can prevent connections (and thus their interface when that connection is active) from ever getting the default route by setting this option.

With nmcli:

```
nmcli c mod eth0 ipv4.never-default true
nmcli c mod eth0 ipv6.never-default true
```

would prevent connection "eth0" from ever receiving the IPv4 or IPv6 default route. This would allow some other connection/interface to receive the default route, when active.

Example:

```
root@rhce1 ~]# nmcli con mod hostonly ipv4.never-default true
[root@rhce1 ~]# ip r
default via 10.23.23.99 dev enp0s8  proto static  metric 1024
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
10.23.23.0/24 dev enp0s8  proto kernel  scope link  src 10.23.23.51
10.23.23.0/24 dev veteam0  proto kernel  scope link  src 10.23.23.104

root@rhce1 ~]# systemctl restart network.service

[root@rhce1 ~]# ip r
default via 10.0.2.2 dev enp0s3  proto static  metric 1024
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
10.23.23.0/24 dev enp0s8  proto kernel  scope link  src 10.23.23.51
10.23.23.0/24 dev veteam0  proto kernel  scope link  src 10.23.23.104
```

5. Firewall with firewalld

Usefull resources are

- certdepot.net [http://www.certdepot.net/rhel7-get-started-firewalld/]
- Redhat [https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html]
- Fedora [https://fedoraproject.org/wiki/FirewallD]

5.1. General

The systemwide default zone is configured in

```
/etc/firewalld/firewalld.conf
```

and can be changed either with an editor, or with

```
firewall-cmd --set-default-zone=<desired default zone>
```

5.2. Quest: create a custom firewall zone

The task we are persuing in order to learn about firewalld is to put team0 in an extra zone and

- block ssh
- allow http/ or port 1234
- for all clients / allow for only one client

Note

When opening a nonstandard listener with nc, issue "setenforce 0" to put the server in permissive mode, otherwise SELinux will block access to the listener, as there is no policy allowing that listener.

Create a new zone by copying one of the standard zones, e.g.:

```
[root@rhce1 ~]# cd /etc/firewalld/zones/
[root@rhce1 zones]# cp /usr/lib/firewalld/zones/drop.xml team.xml
[root@rhce1 zones]# vim team.xml
[root@rhce1 zones]# cat team.xml
<?xml version="1.0" encoding="utf-8"?>
<zone target="DROP">
  <short>team</short>
  <description>Special Firewall zone for my team interface.</description>
</zone>
```

Now you could go on editing the xml file with an editor to add services, but this could also be done using firewall-cmd. First, we check the active zones:

```
[root@rhce1 zones]# firewall-cmd --get-active-zones
```

```
public
  interfaces: enp0s3 enp0s8 enp0s9 veteam0
```

Then, we add the device to the zone:

```
[root@rhce1 ~]# nmcli connection modify team0 connection.zone team
[root@rhce1 ~]# nmcli connection show team0 |grep zone
connection.zone:                        team
```

Verify with firewall-cmd that team0 was added to the zone team which is now active:

```
[root@rhce1 zones]# firewall-cmd --get-active-zones
public
  interfaces: enp0s3 enp0s8 enp0s9
team
  interfaces: veteam0
```

Now we need to add our custom port to the zone team. First the code, then the explanation:

```
[root@rhce1 zones]# firewall-cmd --zone=team --list-ports
[root@rhce1 zones]# firewall-cmd --zone=team --add-port=1234/tcp --permanent
success
[root@rhce1 zones]# firewall-cmd --zone=team --list-ports
[root@rhce1 zones]# firewall-cmd --reload
success
[root@rhce1 zones]# firewall-cmd --zone=team --list-ports
1234/tcp
```

First we check the open ports of the zone "team" - no ports open. Then we add our custom port and make it permanent. The we check again - still no ports open. A reload is required to make the changes effective (without reboot).

Now with selinux in permissive mode, I start a custom listener like this:

```
nc -l 1234
```

Now I *should* be able to connect form the outside to that port. I am not, the remainder of this section contains debugging trials.

I can not connect to that port from the outside (telnet 10.23.23.77 1234) with the firewall active. When I put team0 in the public zone and add the custom port, it somehow works:

```
[root@rhce1 ~]# nmcli connection modify team0 connection.zone public
[root@rhce1 zones]# firewall-cmd --zone=public --add-port=1234/udp --permanent
success
[root@rhce1 zones]# firewall-cmd --zone=public --add-port=1234/tcp --permanent
success
[root@rhce1 zones]# cat public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks
  <service name="dhcpv6-client"/>
  <service name="ssh"/>
  <port protocol="tcp" port="1234"/>
  <port protocol="udp" port="1234"/>
</zone>
[root@rhce1 zones]# firewall-cmd --reload
```

```
success
```

So there must be a subtle difference in the way the zones are configured. Maybe after removing the "DROP" from the team config, a complete firewall reload is needed:

```
firewall-cmd --complete-reload
```

No, that does not help either.

Also, ssh should be blocked for team0, but is not. It kind of looks like only the default zone is active, maybe because both interfaces have addresses in the same subnet. So next steps are to create another hostonly subnet, put team0 (or whatever interface) in that subnet and try the same exercise again.

Other try: Query whether interface interface is bound to a zone.

```
[root@rhce1 ~]# firewall-cmd --get-active-zones
public
  interfaces: enp0s3 enp0s8 enp0s9
block
  interfaces: veteam0
  sources: 10.23.23.11
[root@rhce1 ~]# man firewall-cmd
[root@rhce1 ~]# firewall-cmd --zone=block --query-interface=veteam0
yes
[root@rhce1 ~]# firewall-cmd --zone=public --query-interface=enp0s9
yes
```

Still, it looks like the assignment to zones does not work, as connecting to port 22 always works, regardless of the zone the interface is in.

Further question:

- Q: If I add a port to a zone, does that apply to incoming or outgoing traffic? A: It applys to incoming traffic (CHAIN INPUT), listed ports will be allowed.
- Q: If I add a source, does that exclude or include that source from traffic? A: If I add a source, all traffic from that source will be blocked.
- What are the defaults for new zones? Like drop/reject, etc.

Answers: By default, all ports are blocked. Guesses: if I add a source, only this source will be allowed ports and services which defined for that zone.

Simpler Quest: create a network connection, which does not allow port 22. Adding interface enp0s10 to zone drop and reloading firewall does not block port 22. Obviously, I missed something. Strange enough, after sleep of laptop, now it works. Check again.

Also, play around with issuing the same command with and without "--permanent", to change both runtime and permanent configuration. Also, first dropping everything, then opening port 22 does not open port 22, but adding service ssh does. Check again. Maybe also wait a little. Also, try defining custom port as a service first (/usr/lib/firewalld/services)

Once again from start:

```
[root@rhce1 ~]# firewall-cmd --get-active-zones
```

```
drop
  interfaces: enp0s10
public
  interfaces: enp0s3 enp0s8 veteam0
[root@rhce1 ~]# firewall-cmd --zone=drop --list-all
drop (active)
  interfaces: enp0s10
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

```
> Port 22 is blocked for enp0s10
```

Now adding ssh: [root@rhce1 [mailto:root@rhce1] ~]# firewall-cmd --zone=drop --list-all drop (active) interfaces: enp0s10 sources: services: ssh ports: masquerade: no forward-ports: icmp-blocks: rich rules:

```
> port 22 is still blocked
```

And then sometimes, it just works. It seems this behaviour is highly undeterministic - there might be a bug. Either in the OS, or maybe the virtual box virtual network is messing with us.

6. Kernel parameters

Nothing here yet.

TODO

7. authconfig

Nothing here yet.

TODO

8. iscsi

8.1. On the server

iscsi targets are managed with `targetcli` (contained in the package by the same name). In this example, we will add a second hard disk to our iscsi server ("target"), which is available under the name of `/dev/sdb`. The following sequence of commands is taken from the manpage of `targetcli`, where it is presented as a quick start, most of the explaning text is also copy/paste from the manpage (horray for manpages!).

First we must create a backstore, which is a store that "backs" whatever will be exported. Different types are available, among these are FILEIO (usually a local disk image file, but can also be used to create a buffered block device) and BLOCK (for serving a block device which is attached locally).

```
backstores/block create my_disk /dev/sdb
```

Create an iSCSI target with a default WWN. It will also create an initial target portal group called tpg1.

```
iscsi/ create
Created target iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.dcefd2a66602
Created TPG 1.
```

Add a portal, i.e. an IP address and TCP port via which the target can be contacted by initiators. Sane defaults are used if these are not specified.

```
cd iscsi/iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.dcefd2a66602/tpg1
portals/ create
```

Create ACLs - who can access the lun? First look up your iscsi ID on the client:

```
[root@rhce1 ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:8445b9e15b84
```

Then, back on the server:

```
acls/ create iqn.1994-05.com.redhat:8445b9e15b84
```

Create a new LUN in the TPG, attached to the storage object that has previously been defined. The storage object now shows up under the /back# stores configuration node as activated. Also, disable authentication, generate node acls (whatever that means …), and save the configuration:

```
luns/ create /backstores/block/my_disk
set attribute authentication=0
set attribute generate_node_acls=1
cd /
saveconfig
```

The last two steps are not really nessessary, if you just exit wit `exit` the config will also be saved.

In `targetcli`, you can always type `help` to get a list of commands to navigate. Using the tab key, you will be presented with a list of currently (that is, with respect to your current location in the config tree) available commands. It will also show possible options to commands that you have allready started typing - much like bash completion for e.g. nmcli.

In case you messed up your config, and want to start from scratch, you can use

```
targetcli clearconfig confirm = true
```

We will also need to open the corresponding firewall port

```
firewall-cmd --add-port=3260/tcp
firewall-cmd --add-port=3260/tcp --permanent
firewall-cmd --reload
```

If you forget about which ports are used by default, you can always look them up:

```
[root@rhce2 ~]# grep iscsi-target /etc/services
iscsi-target    3260/tcp                # iSCSI port
iscsi-target    3260/udp                # iSCSI port
```

8.2. On the client

First, install the client utilities:

```
yum install iscsi-initiator-utils
```

Then discover the targets with `iscsiadm` :

```
[root@rhce1 ~]# man iscsiadm
[root@rhce1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 10.23.23.52 --di
10.23.23.52:3260,1 iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.a0504e72987a
```

The man command is shown as the whole command is listed in the man page, so no need to remeber all the options. Note that the ip address is resolved to rhce2 - it is important to have dns working. This can be via `/etc/hosts`.

Now we need to "login" to the target. The identifier used must be the one found by the previous discovery action. The exact syntax to login is written in the man page of `iscsiadm`. We use `lsblk` bevor and after the login to show that we have a new device:

```
[root@rhce1 ~]# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                           8:0    0    8G  0 disk
  sda1                        8:1    0  500M  0 part /boot
  sda2                        8:2    0  7.5G  0 part
    centos_centosseven-swap 253:0    0  820M  0 lvm  [SWAP]
    centos_centosseven-root 253:1    0  6.7G  0 lvm  /
sr0                          11:0    1 55.6M  0 rom
[root@rhce1 ~]# iscsiadm --mode node --targetname \
    iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.a0504e72987a \
    --portal 10.23.23.52:3260 --login
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.a0504e
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.rhce2.x8664:sn.a0504e72987
[root@rhce1 ~]# lsblk
NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                           8:0    0    8G  0 disk
  sda1                        8:1    0  500M  0 part /boot
  sda2                        8:2    0  7.5G  0 part
    centos_centosseven-swap 253:0    0  820M  0 lvm  [SWAP]
    centos_centosseven-root 253:1    0  6.7G  0 lvm  /
sdb                           8:16   0    8G  0 disk
sr0                          11:0    1 55.6M  0 rom
```

Now we can use the new device `/dev/sdb` for whatever we want. If, after we created a filesystem, want to make the mount permanent, we must take care to add the option `_netdev` to `/etc/fstab`, which tells mount that this device requires working networking. So the mount is delayed untill the network is available.

9. System Activity Accounting

Various aspects of system activity (cpu, ram, device throughput, …) can be monitored with `sar`. `sar` collects the data and writes it to daily binary reports under `/var/log/sa/`.

`sar` is contained in the `sysstat` package. Without any options, `sar` displays cpu activity. Consult the man page for the myriard of options available.

The frequency of measurement is controlled by the cronjob defined in `/etc/cron.d/sysstat`. Other parameters (e.g. how many files to keep) are stored in `/etc/sysconfig/sysstat`.

Another tool contained in the `sysstat` package is `iostat`. It can be called like this:

```
iostat <intervall lenth in seconds> <number of interfalls to display>
```

E.g.

```
iostat 2 5
```

Without the number of intervalls, new data is displayed for ever.

10. Shell scripting

There are a lot of great resources on the web about shell scripting, so here I will only list bits and pieces that I was no aware of or did not know by heart.

10.1. Special Variables

Some special variables $* - all parameter in one string $@ - all parameter in one string, each parameter quoted on its own. $# - number of parameters given to a script or function

10.2. Case

A switch case construct is also available in bash, though I seldom use it:

```
case $1 in
    foo) echo You entered foo ;;
    ba[rz]) echo You entered something starting with ba, then r or z. ;;
    *) echo I do not recognize this ;;
esac
```

A point I was not aware of is the use of pathname expansion used in the cases.

10.3. String manipulation

All sorts of string manipulation can be done in bash. Check `man bash` for a comprehensive read. I was amazed to see what is possible.

Anyway, here are a few examples of string manipulation in bash:

```
my_string="Hello World out there"
echo "my_string=${my_string}"
echo "Length of my_string is ${#my_string}"
echo "Substring, starting at 3, 5 chars long: ${my_string:3:5}"
echo "Matches are done according to the rules of filename expansion"
echo "removing prefix up to match of o, lazy: ${my_string#*o}"
echo "removing prefix up to match of o, greedy: ${my_string##*o}"
echo "removing postfix from match of o, lazy: ${my_string%o*}"
echo "removing postfix from match of o, greedy: ${my_string%%o*}"
echo "search and replace: ${my_string/World/Folks}"
```

There is more you can do with variables in bash, have a look at the man page, its pretty well written.

10.4. Loops

All sorts of loops are availabe. Of the usefull ones, this is the one I always forget about:

```
while read i; do
    echo $i
done < myfile.txt
```

This reads the content of the file `myfile.txt` line by line.

10.5. Miscellaneous

Date calculations relative to the current date can be done like this:

```
date -d "+12 weeks + 2 hours + 23 minutes"
```

Write to stderr:

```
echo This is an Error >&2
```

You can create a temporary file or directory with `mktemp`. This could be usefull if you need a temporary file with a uniq name - `mktemp` will take care of this:

```
[root@rhce1 ~]# x=$(mktemp /tmp/foobar.XXXX)
[root@rhce1 ~]# echo $x
/tmp/foobar.ZlZq
[root@rhce1 ~]# echo Hi >$x
[root@rhce1 ~]# cat $x
Hi
```

Get a (pseude) random number:

```
echo $RANDOM
```

Thats all, folks.

11. Installing Software

Software is installed using yum. It might be worth installing `yum-utils`, which enhances yums capablities.

If you would like to have a persistent cache of the metadata, you can tell yum to keep the cache by setting

```
keepcache=1
```

in `/etc/yum.conf`. After that you should clean and rebuild the cache:

```
yum clean all
yum makecache
```

Then you might want to have a cron job which keeps your cache up to date (e.g. `yum-updatesd`).

12. SELinux

To manage various aspects of SELinux, a tool called `semanage` is needed. It is not installed by default, even though SELinux is active by default. I did not know which package contained `semanage`, so I asked yum:

```
yum whatprovides semanage
yum install policycoreutils-python selinux-policy-devel
```

The second package is need for the manpages - this is something which you need to **remember by heart**.

Various resources like files, directories, network ports (!) and processes have labels, which control which object can access which resource (or in which way). Each label has four components:

- user component (usually ends with _u)
- role component (usually ends with _r)
- type component (usually ends with _t)
- multi level security (MLS) component.

I will not go into detail here, as it can become almost arbitrarily complex. The kernel manages the enforcement of the rules.

If you copy a file, it will inherit its new label from its parent directory. If you move a file, it will keep its label, and you will probably need to run `restorecon` to apply the proper label to it.

The rules guiding the labeling are stored in txt files underneath `/etc/selinux/targeted/contexts/`. As you install new software with yum, these files will be updated.

To easily search for labels containing certain strings, you can use `semanage` (also ports):

```
[root@rhce1 ~]# semanage port --list |grep ssh
ssh_port_t                     tcp      22
[root@rhce1 ~]# semanage fcontext --list |grep ssh
/etc/rc\.d/init\.d/sshd                            regular file       system_u:object_r:
/etc/ssh/primes                                    regular file       system_u:object_r:
/etc/ssh/ssh_host.*_key                            regular file       system_u:object_r:
[...]
```

And to get a list of all possible contexts, use

```
seinfo -t
```

and then grep for what you are looking for, e.g.

```
seinfo -t |grep samba
```

to get all samba related contexts.

12.1. Policys

The default policy is "strict", which means everything is denied except for explicitly allowed actions. This seems to be not so managebale in practise. Therefore, usually a "targeted" policy is used. This only restricts a few components, and allows the rest. In the "targeted" policy, only the type component of the label is checked.

12.2. Booleans

Certain settings of a policy are controlled by booleans, which can be toggeled at runtime. A list of available booleans can be viewed with

```
getsebool -a
```

To set a boolean, use

```
setsebool -P <boolname> <value>
```

The `-P` option makes the change permanent, without it the change in bool setting will not survive a reboot.

12.3. Audit

Policy violations (and also success) are logged in `/var/log/audit/audit.log`. So if something does not work, you can check there. In principal, everything is said there, but it is not the most human frindly format, there a tool has been defeloped, which can translate:

```
audit2why </var/log/audit/audit.log
```

If you do not want to see the whole log, you can extract the line in question to a different file and pipe that to `audit2why`.

You can also use `sealert` to generate concrete semanage commads to fix your selinux problem. E.g. you want sshd to llisten also on port 2222. After configuring it, sshd won’t start port 2222, but you get a message in the audit log. This cna be translated to a fix like this:

```
sealert -a /var/log/audit/audit.log
```

This gives lots of output, among others:

```
# semanage port -a -t PORT_TYPE -p tcp 2222
     where PORT_TYPE is one of the following: ssh_port_t, vnc_port_t, xserver_port_t.
```

So you just execute the command given, and then sshd can listen on port 2222.

12.4. SELinux in practise

If SELinux denies access to a file because of the wrong context (e.g. you have moved your .ssh/ authorized_keys from somewhere else instead of creating it in place), you can restore the defined file context with `restorecon`:

```
restorecon -rv /home/isaac/.ssh
```

This of course relies on files being in their standard location. In practise this is not always the case. E.g. you might have your http doc root not in `/var/www/html`, but in `/opt/www/htmp`. `restorecon` won't help much here, since it has no context for `/opt/www` yet. But you can create that context with `semanage` (`man semanage-fcontext`), provided you know which context you want. For this it is helpfull to create a file in the standard location and check which context gets applied to this file. In this case, it would be `httpd_sys_content_t`.

```
semanage fcontext -a -t httpd_sys_content_t "/opt/www(/.*)?"
```

This specific regualr expression is important, and it should be explained(???). The manpage for this particular subcommand can be read with `man semanage-fcontext`

A list of examples for the regular expression can be optained with e.g.

```
semanage fcontext -l |grep "/var/www"
```

A different approach is to take an existing directory with the correct context and use that as a template:

```
semanage fcontext -a -e /var/www/html /opt/www/html
```

Both add new file context rules, which can then be applied with

```
restorecon -rv /opt/www/html
```

Example:

```
[root@rhce1 html]# ls -lZ
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 index.html
[root@rhce1 html]# semanage fcontext -a -e /var/www/html /opt/www/html
[root@rhce1 html]# restorecon -rv /opt/www/html/
restorecon reset /opt/www/html context unconfined_u:object_r:usr_t:s0->unconfined_u:obje
restorecon reset /opt/www/html/index.html context unconfined_u:object_r:user_tmp_t:s0->u
[root@rhce1 html]# ls -lZ
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 index.html
```

If you want to run your service on a non-standard port, you need to add a suitable label to that port:

```
semanage port -a -t http_port_t -p tcp 8080
```

This labels tcp port 8080 for use by httpd.

12.5. Create your own SELinux module

If there is no suitable file or port context for your needs, you can create your own packge, using `audit2allow`.

```
audit2allow --input=myaudit.log --module-package=myaudit
semange --install myaudit.pp
```

`audit2allow` will create a clear text file with .te ending, and a binary module with .pp ending. The binary can be installed with semanage. Note that the `-i|--install` option is not documented in the manpages.

EXERCISE: do this httpd with custom doc root

12.6. Changing file context

File contexts can be changed on the fly with e.g.

```
chcon -t public_content_t /nfsshare
```

13. Apache httpd

13.1. Basic virtual hosts

An example for a basic virtual host config is given in `/usr/share/doc/httpd-2.4.6/httpd-vhosts.conf` (see rpm -qd httpd):

```
<VirtualHost *:@@Port@@>
    ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot "@@ServerRoot@@/docs/dummy-host.example.com"
    ServerName dummy-host.example.com
    ServerAlias www.dummy-host.example.com
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

Note that `@@Port@@` should be replaced with the actual port httpd is listening on, usually 80 or 443. These configuration lines should be placed in a file in `/etc/httpd/conf.d` with the ending `.conf`.

13.2. https

Installing `mod_ssl` also adds a default ssl config, which uses the self signed certificats which are already present on the server. These can be found in `/etc/pki/tls/certs`.

https works out of the box, installing mod_ssl also puts a default `/etc/httpd/conf.d/ssl.conf` ssl config on your system. It is heavly documented, so probably no need to learn much by heart. But if you want to customize your installation, e.g. with your own certificate, the following options could be helpfull:

```
SSLEngine on
SSLProtocol All -SSLv2 -SSLv3
SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA2
SSLHonorCipherOrder on
SSLCertificateFile /etc/pki/tls/certs/mycrt.crt
SSLCertificateKeyFile /etc/pki/tls/private/mycrt.key
SSLCertificateChainFile /etc/pki/tls/certs/myca.crt
```

Note that we are disableing SSLv2 and SSLv3, as these have been shown to be vulnerable by the poodle attack.

The complete apache manual is available as the `httpd-manual` package, which will install all of the official apache docs in `/usr/share/httpd/manual`. Theses are html files, so its probably nicer to view them with a browser. It contains all the possible configuration options.

If you want to automatically rewrite all http traffic to https, use the following directive:

```
RewriteEngine on
```

```
RewriteRule ^(/.*)$ https://%{HTTP_HOST}$1 [redirect=301]
```

It looks daunting to learn this by heart, but with a little understanding of regular expressions, it is not that hard: First we look at the url (e.g. www.example.com/foo/bar) from the beginning (`^`) to the end (`$`). We want to extract the part after the hostname, which starts with a slash `/`. Everything behind that is matched with `.*` (`.` matches any character, and `*` matches any number of charachters). Since we want to insert the matched part in the rewrite again, we need to enclose it in brackets `(/.*)` (this is called "numbered backreference").

So that is the matching of incomming url. The next part is what the incomming url is rewritten to. It starts with `https://`, since the whole point of the exercise is to rewrite all traffic to https. Then comes the host name (obvious), and the comes `$1`, which is a reference to the machted expression from the incomming url. That is the expression matched previously within the brackets. In our example this is `/foo/bar`.

The last part is the http status message that the server sends to the client, in this case `[redirect=301]`.

If you insert this line in e.g. in your virtual host block for http traffic, all will be redirected to https.

13.3. apache and php

To make apache able to run php script, you need to install the `php` package. Then you can have php commands in side the following tag:

```
<?php
    //php command, e.g.
    phpinfo();
?>
```

This must be inside a file with an `.php` extension. The rest of the file can still contain html.

13.4. apache and python

To make apache execute python scripts, install `mod_wsgi`. That is almost enough to remember, the rest is in the docs that come along with the package. To find the docs, do something like

```
grep -Ri wsgi /usr/share
```

This will turn up `/usr/share/doc/mod_wsgi-3.4/README`, which contains everything needed for a basic setup. It boils down to the following:

Inside your virtualhost directive, add this line to tell apache what to do with a certain url:

```
WSGIScriptAlias /py /usr/local/apache/mypy.py
```

Note that the python script resides outsite of the normal DocumentRoot. This is on purpose, as it might enable others to download the script itself if it would reside inside the doc root. But this also means that the script needs to be accessible and executable by the apache user. And because we have not set any specific restrictions on that dir yet, apache will deny everything, so we will need to allow access:

```
<Directory "/usr/local/apache">
```

```
        Require all granted
</Directory>
```

Next is to deploy a valid wsgi application. The README also provides a working example, which can be copied:

```
def application(environ, start_response):
    status = '200 OK'
    output = 'Hello World!'
    response_headers = [('Content-type', 'text/plain'),
                        ('Content-Length', str(len(output)))]
    start_response(status, response_headers)
    return [output]
```

The important part is that it contains a function by the name of *application*.

14. DNS

There are two packages for doing DNS that will be examined here: bind and unbound.

14.1. Caching only name server with bind

bind is configured via `/etc/named.conf`. `named` is also the name of the service. Details about the config can be found with `man named.conf`. Also, the default config file lists quite a few options. For a caching name server, the following directives are needed:

```
listen-on port 53 {any;};   // any could also be a semicolon seperated list of ip addres
allow-query {localhost; 10.23.23.0/24 };
recursion yes;
forward only;
forwarders {8.8.8.8; };
```

So here we define a DNS server that allows recursion (which makes sense since it is caching only), and that forwards requests. A list of DNS servers is defined which the queries are forwarded to (`forwarders`), we listen on port 53 on all interfaces, and allow queries from localhost and the `10.23.23.0/24` subnet. If everyone should be able to access the server, then the `allow-query` directive can be ommited.

Now bevor starting the servic,e we can check the config for syntax errors:

```
named-checkconf
```

If that did not show any errors, we can start named.

You can get an answer from a specific server e.g. like this:

```
dig @localhost www.google.com
```

14.2. Caching only name server with unbound

unbound is configured via `/etc/unbound/unbound.conf`. The default file is commented, so lots of options are already explained there. A minimal caching only nameserer would be configured like this:

```
server:
        interface: 0.0.0.0
        access-control: 10.23.23.0/24 allow
        domain-insecure: "example.com"
remote-control:
        control-enable: yes
forward-zone:
        name: "."
        forward-host: 8.8.8.8
```

The option `domain-insecure` specifies domains where validation should be ommited. This might be interesting for local zones.

14.3. DNS-Troubleshooting

Maybe the best tool for dns troubleshooting is `dig` (check `yum whatprovices dig` for the package it is included with).

The order in which dns queries are done (local, dns server, …) is defined in `/etc/nsswitch`, in the line that starts with `hosts`. In order to check if your system is ok, you can compare the output of

```
dig @yournameserver www.google.com
getent hosts www.google.de
```

`getent` respects `/etc/nsswitch`, so a difference in the output could hint to a problem with the local system.

You could also force dig to use tcp (if udp is somehow blocked):

```
dig +tcp www.google.com
```

15. NFS

15.1. Setting up a kerberos server

One of the requirements for the RHCE is to authenticate various services using kerberos. Setting up a kerberos server itself is not part of the exam, but for preparational purposes, I set one up using the following guide: https://gist.github.com/ashrithr/4767927948eca70845db [kerberos basic setup].

I had to do a few things which are not mentioned in the original guide. Most of them are probably obvious, but I still mention them here for completness.

Add the kerberos service to the firewall:

```
[root@rhce2 services]# firewall-cmd --permanent --add-service=kerberos
success
[root@rhce2 ~]# firewall-cmd --reload
success
```

Define a new service for the kerberos admin service (see also http://b.minwi.com/2014/09/11/define-a-new-service-in-firewalld/):

```
[root@rhce2 services]# cat /etc/firewalld/services/kerberos-adm.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
```

```
  <short>Kerberos-adm</short>
  <description>Kerberos-adm network authentication protocol server</description>
  <port protocol="tcp" port="749"/>
  <port protocol="udp" port="749"/>
</service>
```

Not the port 749. Then add that service to the firewall:

```
[root@rhce2 services]# firewall-cmd --permanent --add-service=kerberos-adm
success
[root@rhce2 services]# firewall-cmd --reload
success
```

Also there is a typo, the command to add a principal on the client should be

```
addprinc -randkey host/client.example.com
```

15.2. Exporting directories ro and rw with no auth

Exporting a direcorty without authentication (only IP address of client is checked if desired) is rather simple. Install `nfs-utils`, start and enable the `nfs` service, and add it to the firewall.

A simple `/etc/exports` could look like this

```
/export *(ro)
```

This allows everybody to mount `/export` read only. If the `*` is replaced with an IP-address, or a subnet, then only clients with this adress or from this subnet can mount.

In order to export a directory writable, you have two possibilities:

- make the exported dir owned by `nfsnobody` (and use `root_squash` as an option for the export)
- make the exported dir owned and mounted by root, and use the option `no_root_squahs`.

Because the latter has some security implications, I would use the former. The option `root_squash` maps files which would belong to the root user on a client to the user "nfsnobody".

In order to make changes to `/etc/fstab` visible to the server, it is not nessessary to restart the server, a simple

```
exportfs -r
```

will do.

On the client (or on the server), you can check which filesystems are exported on a nfs server with

```
showmount -e your.nfs.server
```

Notice that you need to enable `mountd` and `rpc-bind` in the firewall for this to work.

15.3. NFS and Firewall

In order to run an nfs server, you need to enable the following services in the firewall:

- nfs

- mountd
- rpc-bind

The latter two are not absolutly nessessary, but in order to make `showmount` work, they are needed.

15.4. Kerberizing NFS

Check http://www.certdepot.net/rhel7-use-kerberos-control-access-nfs-network-shares/ .

The following is not a complete list, but some of the things I need to remember.

You will need to kerberize your client and server. The important thing is to have your `/etc/krb5.conf` and `/etc/krb5.keytab` on both client and server. In the scope of rhce, these files will be provided.

On the server: In `/etc/sysconfig/nfs`, in order to export SELinux labels, make sure to have

```
RPCNFSDARGS="-V 4.2"
```

as previous versions of nfs did not support this. Then start and enable **nfs-secure-server**. Then, in your `/etc/exports`, have something like

```
/securenfs clientfoobar(sec=krb5p,rw)
```

Here the interesting part ist the `sec=krb5p`. This offers authentication, encryption and integrity via kerberos. Other options would be `krb5i`, which would be authentication and integrity, and `krb5`, which would be authentication only.

On the client:

Start and enable **nfs-secure** on the client. Then add something like the following to your `/etc/fstab`:

```
rhce1:/securenfs    /mnt/nfs    nfs defaults,v4.2,sec=krb5p 0 0
```

It might be nessessary to modify the SELinux content of exported files or dirs:

```
chchon -t public_content_t /securenfs/mytestfile.txt
```

15.5. General NFS Options

Here are some general NFS Options:

- `ro` read only
- `rw` read write
- `sync` make sure all writes are synchronized to disk bevor acknowledging a successfull write. This improves data security, but costs (quite a lot of) performance
- `root_squash` All files created on the client by root will have its ownership mapped to a less privileged user. This is the default. The opposite would be `no_root_squash`, which is less advisable from a security point of view.

- `insecure` client requests are allowed to come from an unprivileged port (meaning normal users can mount). The default is the oposite, which is `secure`.
- `nohide` Show mounted directories inside the export. The default is `hide`.

16. Samba

16.1. Basics

Install the samba server by installing the packages `samba` and `samba-client`. This gives you a well documented `/etc/samba/smb.conf`.

By default, it exports homedirs and printers. Make sure to read the section about SELinux at the top of the default configuratin file. Then enable samba in the firewall, start the service and enable home dir sharing in SELinux:

```
[root@rhce2 ~]# vim /etc/samba/smb.conf
[root@rhce2 ~]# firewall-cmd --permanent --add-service=samba
success
[root@rhce2 ~]# firewall-cmd --reload
success
[root@rhce2 ~]# systemctl start smb nmb
[root@rhce2 ~]# setsebool -P samba_enable_home_dirs on
```

If you want to export a different direcory, you could for example use the provided `[public]` share:

```
# A publicly accessible directory that is read only, except for users in the
# "staff" group (which have write permissions):
        [public]
        comment = Public Stuff
        path = /home/samba
        public = yes
        writable = yes
        printable = no
        write list = @staff
```

Note that the samba users - which are mapped to system users - need to have permissions on the directories that we want to export. So if I have a systemuser `isaac`, I need to add that user also to samba (`smbpasswd -a isaac`), and then give read (and write) permission to isaac on the exported dir. This can be acomplished in different kinds of ways, either through classic filesystem permissions, or with ACLs, which is much more flexible and probably what you want in this case.

If the samba share is owned by a specific group, it is advisable to give the SGID bit on the parent directory:

```
chgrp staff /home/samba
chmod 2775 /home/samba
```

To see all possible configuration items, use

```
testparm -v
```

Without the `-v`, only currently set items are shown.

16.2. Mounting a samba share on a client

To mount a samba share on an client, you issue something like this:

```
mount -t cifs //10.23.23.52/public /mnt/smb/ -o username=isaac,password=xxxxx
```

Note that this requires the package `cifs_utils` (it provides mount.cifs). Also note that the syntax for specfying a share is

```
//<hostname or IP>/<share name>
```

where `<share name>` is NOT the path on the server, but the name of the share, as specified in square brackets, see above.

If you want to mount remote samba shares as homedirs, you need to toggle the following SELinux boolean:

```
setsebool -P use_samba_home_dirsi=on
```

This is not to be confused with `samba_enable_home_dirs`, which controls the exporting of homedrs.

16.3. Multiuser mounts

To mount a share with multiuser functionality, user a fstab entry like this:

```
//rhce1/sharename cifs credentials=/root/smbcred.txt,multiuser,sec=ntlmssp 0 0
```

16.4. Tools

You can get a listing of all samba users with

```
pdbedit --list
```

There is a tool to edit the samba passwords:

```
smbpasswd -a <username>
```

This will add a user to the local samba passwd file.

17. Postfix

17.1. master.cf

After installing the postfix package, you can find some default configuration files in `/etc/postfix`. There is the `master.cf`, which controls the master process. For the scope of what is expected for RHCE, I expect no modifications are needed to be made there.

17.2. main.cf

And then there is the `main.cf`, where most of the configuration occurs. By default, the system can relay from local recepients to local recepients.

You need to make sure the hostname is set correctly, with fqdn. If you want to relay mails from your local network, check the following two parameters:

```
mynetworks = 10.23.23.0/24, 127.0.0.1 # usually the default
inet_interfaces = all
```

`mynetworks` specifies a range of hosts, which are trusted automatically, which means that mails from them will be accepted without any further checks.

`inet_interfaces` specifies on which interfaces to listen to.

There are other parameters that need to be set, but most of them have good defaults, and in order to do local mail delivery, everything is already set up.

17.3. Mailrelay

If postfix is not only to deliver to local mailboxes, but forward mails that it can not deliver localy, a mail realy needs to be configured. This is done in `main.cf` with the parameter

```
relayhost = rhce1.acme.ch
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
```

for example. After reloading postix, all mail that can not be delivered locally will be forwarded to relayhost. Note that `mydestination` needed the appendix of `$mydomain`, otherwise postfix would not accept mails for other hosts in my domain.

17.4. Aliases

Basic alias configuration is done in

```
/etc/aliases
```

which is a simple text file. It defines aliases for local mail delivery. Check the contents of the file - it is already populated with defaults. After you make changes to this file, it is important to issue

```
newaliases
```

which reads the textfile and transforms it into a binary format (`/etc/aliases.db`), which will then be read by postfix. This does not require a restart of the postfix daemon.

17.5. Tools

To check the actual setting of a parameter, it is helpfull to use the `postconf` utility, which by default displays all parameters and their current setting. For example, to view the current `mynetworks` setting, type

```
postconf |grep mynetworks
```

In order to do basic testing, it can be helpfull to know the basic smtp commands. Connect to port 25 of your mail relay, then issue

```
root@rhce1 ~]# telnet rhce2 25
Trying 10.23.23.52...
```

```
Connected to rhce2.
Escape character is '^]'.
220 rhce2.acme.ch ESMTP Postfix
HELO rhce1.acme.ch
250 rhce2.acme.ch
MAIL FROM:isaac@acme.ch
250 2.1.0 Ok
RCPT TO:root@rhce2.acme.ch
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello root,

how do you do?

yours,
Isaac
.
250 2.0.0 Ok: queued as AE62E90CAF8
```

The basic commands to remember are:

```
HELO <fqdn local host>
MAIL FROM:<sender mail address>
RCPT TO:<recipient mail address>
DATA
. # finish with period, followed by a blank line.
```

Each command is answered with a status code. 250 means "OK", 354 means "Start mail input; end with <CRLF>.<CRLF>". Then, on rhce2, we are informed that mail was received:

```
[root@rhce2 ~]#
You have mail in /var/spool/mail/root
[root@rhce2 ~]# cat /var/spool/mail/root
From isaac@acme.ch  Wed Mar  4 01:06:38 2015
Return-Path: <isaac@acme.ch>
X-Original-To: root@rhce2.acme.ch
Delivered-To: root@rhce2.acme.ch
Received: from rhce1.acme.ch (rhce1 [10.23.23.51])
        by rhce2.acme.ch (Postfix) with SMTP id AE62E90CAF8
        for <root@rhce2.acme.ch>; Wed,  4 Mar 2015 01:05:54 -0500 (EST)

Hello root,

how do you do?

yours,
Isaac
```

So things are working. If not, check `/var/log/maillog`. Each mail gets its uniq ID, which is the last part of the conversation of the mta with the client (`250 2.0.0 Ok: queued as AE62E90CAF8`). We can grep for that ID to see what happend with that email:

```
[root@rhce2 ~]# grep AE62E90CAF8 /var/log/maillog
Mar  4 01:06:18 rhce2 postfix/smtpd[2734]: AE62E90CAF8: client=rhce1[10.23.23.51]
Mar  4 01:06:38 rhce2 postfix/cleanup[2737]: AE62E90CAF8: message-id=<>
Mar  4 01:06:38 rhce2 postfix/qmgr[2705]: AE62E90CAF8: from=<isaac@acme.ch>, size=227, n
Mar  4 01:06:38 rhce2 postfix/local[2738]: AE62E90CAF8: to=<root@rhce2.acme.ch>, relay=l
Mar  4 01:06:38 rhce2 postfix/qmgr[2705]: AE62E90CAF8: removed
[root@rhce2 ~]#
```

In this case we see that themail was queued and delivered to mailbox.

18. SSH

18.1. Keys

Keys are generated with `ssh-keygen`. It has a parameter to specify the "strength" in bytes:

```
ssh-keygen -b 4096
```

The default for RHEL 7 /Centos 7 is 2048 for RSA keys. RSA keys are the default.

18.2. Various

All sorts of other things can be done with the ssh client and server. Extensive documentation can be found in the man pages:

```
man sshd_config # server
man ssh_config # client
```

19. NTP

There are two tools to synchronize time over network: the classic ntp, and the new chrony.

19.1. ntpdate and ntpd

If you want to manually synchronize the time, you can issue something like

```
ntpdate time.server.org
```

where `time.server.org` should be replaced with the fqdn of a real time server. You can also start the service `ntpd`, which will then periodically synchronize time. The downside of this is that ntpd can not correct bigger time differences.

Time servers for ntpd are configured in `/etc/ntp.conf`, where a lot of examples are configured, so the syntax should be clear.

19.2. chrony

The newer chrony, part of systemd, is configured via `/etc/chrony.conf`. There you can define your time servers, the syntax is similar to ntpd - check the file for examples.

chrony runs as a service called chronyd.

20. Maria DB

After installing `mariadb-server`, start the service `mariadb`, and enable it it the firewall. In the firewall it is called `mysql`, because Maria DB is a fork of MySQL.

You can connect to the database locally with

```
mysql -u root
```

20.1. Securing the root Account

The default installatin does not have a password for the local root user. For security reasons, this should be changed. Also, there is more then one root account - accounts have not only a username, but also a hostpart from where the user connects, so e.g. root*@127.0.0.1* and root*@rhce1* are two different accounts, which each can have their own password.

Then there is an guest accout accessible to everyone, intended for testing, and a test database accessible to everyone, also for testing. These should be removed for production use.

This can all be done using sql commands, but is a bit tedious, so there is a script provided which will guide you through all the steps and does the updates to the mariadb for you:

```
mysql_secure_installation
```

This script can also restrict root access from localhost only.

One could go further to rename the root account

20.2. Adding sample data

There is an official sample database, also used for trainings. It is publicly available:

```
wget http://downloads.mysql.com/docs/world_innodb.sql.gz
```

Firt we need to create the database:

```
mysql -u root -p
> create database world;
> exit;
```

Then we load the unzipped data into that database:

```
mysql -u root -p world  </tmp/world_innodb.sql
```

20.3. Navigating a database

Note the syntax: `mysql -u <username> -p <databasename>`. The `-p` specifies to ask for a password, alternativly you could also give the password on the commandline: `--password=xxxxx`, which is less favorable form a security standpoint.

Now login again as root

```
mysql -u root -p
```

and see which databases we have:

```
> show databases;
```

select a database and see which tables it has:

```
> use world;
> show tables;
```

20.4. Creating users

Depending on the setup, it can be good practise to create at least two users per database. One that can only read, and one that can also write and create new tables:

```
> grant all on world.* to 'world-admin'@'10.23.23.%' identified by 'xxxxx';
> grant select on world.* to 'world-user'@'10.23.23.%' identified by 'xxxxx';
> flush privileges;
```

Note the use of the wildcarc `%` in the host identifier - this allows access from the whole subnet. The "flush" command forces mariadb to reread the privileges database.

20.5. CRUD

CRUD is an abreviation for "Create Read Update Delete", which are common actions performed on a database. A Read in MariaDB is done using the "select" statemente, e.g.:

```
mysql -u world-user -p -h rhce1
use world;
select Name from world where CountryCode='deu';
select * from world where CountryCode='deu';
select Name,CountryCode,Population from City where Name like '%ingen%';
select Name,CountryCode,Population from City where 1 order by Population desc limit 3;
select count(*), 'Citys in Germany' from City where CountryCode='deu';
select count(*) as cnt, District from City where CountryCode='deu' group by District \
    order by cnt desc;
```

Create/Insert:

```
insert into City (Name,District,CountryCode,Population) values ('Schoenau', \
    'Baden Wuertemberg', 'deu', '2000');
ERROR 1142 (42000): INSERT command denied to user 'world-user'@'rhce1.lcsys.ch' \
    for table 'City'
mysql -u world-admin -p -h rhce1
use world;
insert into City (Name,District,CountryCode,Population) values ('Schoenau', \
   'Baden Wuertemberg', 'deu', '2000');
```

Update:

```
MariaDB [world]> update City set Population = Population + 4 where Name='Schoenau';
```

Delete:

```
delete from City where Name='Schoenau';
```

20.6. Complex Queries

```
select * from City left join Country on City.CountryCode = Country.Code \
    where City.Name = "Zürich";
```

```
select Country.Name from Country left join CountryLanguage on \
    Country.Code = CountryLanguage.CountryCode where \
    Language = "German" and IsOfficial = "T";
```

21. Cheatsheet

This is no intent to cheat, but rather a fictional cheat sheet. Here I write down bits and pieces that I find hard to remember. By writing them down here, I remember them, so no need to cheat :).

21.1. Samba

Start both smb and nmb.

Mount options for multiuser mount: credentials=/root/credfile.txt,multiuser,sec=ntmlssp

21.2. Kerberized NFS

In `/etc/sysconfig/nfs`, set `RPCNFSDARGS="-V 4.2"`. This allows you to export SELinux contexts.

Mount options: defaults,sec=krb5p,v4.2

21.3. Restting the root password

Boot with boot kernel option rd.break, mount -o remount,rw /sysroot, chroot /sysroot, passwd root, touch /.autorelabel, exit, reboot.

21.4. MariaDB INSERT

```
insert into tablename (col1,col2) values ('val1', 'val2');
```

21.5. SELinux

Get a list of all possible contexts with

```
seinfo -t
```

Get concrete help for selinux problems which are logged to /var/log/audit/audit.log:

```
sealert -a /var/log/audit/audit.log
```

21.6. Apache

```
rpm -qd httpd
-> among others an example virtual host conf
```