1

Task

Configure the system to use the following **NTP pools**:

- `0.europe.pool.ntp.org`
- `1.europe.pool.ntp.org`

Next, change the **timezone** of this system to `Europe, Bucharest`.

Solution

To add the NTP pools to the system, edit the `/etc/systemd/timesyncd.conf` as below:

```
[Time]
NTP=0.europe.pool.ntp.org 1.europe.pool.ntp.org
```

Restart the **systemd-timesyncd** to reflect the chnages:

```
sudo systemctl restart systemd-timesyncd
```

To change timezone of this system to **Europe, Bucharest** run below command:

```
sudo timedatectl set-timezone Europe/Bucharest
```

==========

2

Task

Add a **cron job** for the user called **john**. Don't use the system-wide crontable, but rather add it to the personal crontable of the user called **john**.

Make sure that this cron job runs every **Wednesday at 4AM**. The command it should execute is `find /home/john/ -type d -empty -delete`.

Switch back to the bob user once the task is done.

Solution

1) Switch to user john

```
sudo su john
```

2)Edit the crontab using `crontab -e` command as shown below to run jobs every **Wednesday** at **4 AM**:

```
0 4 * * 3 find /home/john/ -type d -empty -delete
```

==========

3

Task

There is a network interface on this system which has the IP address `10.5.5.2` associated with it. What is the name of this **network interface**? Create a file in `/opt/interface.txt` and add a single line to it containing the exact name of that interface.

Solution

To get the name of network interface which has ip address 10.5.5.2, run below command:

```
ip -o -4 addr list | grep '10.5.5.2' | awk '{print $2}' |sudo  tee /opt/interface.txt
```

==========

4

Task

An administrator added a new user called `jane` to this system. But a few mistakes were made. Fix the following problems:

1. The administrator wanted to allow jane to run sudo commands. But instead of adding "jane" to the secondary/supplemental "sudo" group, the administrator changed the primary group to sudo. Fix this by doing the following: Set the **primary/login group** back to the group called `jane`. And add the user to the **secondary/supplemental group** called `sudo`. In essence the primary group for the user called "jane" should be "jane". And the secondary group should be "sudo".
2. Currently, the **home directory** path for the `jane` user is set correctly. But the directory itself is missing. Fix this by creating the `/home/jane/` directory. Make sure that the directory is owned by the `jane` user and `jane` group.
3. The **default shell** for the user called `jane` is set to **/bin/sh**. Change the default shell to `/bin/bash`.
4. Finally, set the password for `jane` to `1234`.

Solution

1) Change Primary Group of Jane Back to **jane**:

```
sudo usermod -g jane jane
```

2) Add the user **jane** to the secondary group **sudo**:

```
sudo usermod -aG sudo jane
```

3) Create the **Home Directory** for Jane and Set Ownership:

```
sudo mkdir -p /home/jane
sudo chown jane:jane /home/jane
```

4) Change Default Shell to **/bin/bash**:

```
sudo usermod -s /bin/bash jane
```

5)Set the Password for Jane:

```
echo 'jane:1234' | sudo chpasswd
```

==========

5

Task

Perform the following tasks:

1. Set up a port redirection rule that does the following: it **redirects** any **IPv4** packet from the `10.5.5.0/24` CIDR range incoming on port `81` to another machine with the IP address `192.168.5.2` on port `80`. To simplify this task, you are not required to specify input or output network interfaces.
2. Don't forget to add the proper **masquerading rule** so that packets redirected from `10.5.5.0/24` have a way of getting back to that sender, by our machine sitting in the middle and acting as an ad-hoc router between those two networks.
3. Make sure that after you add the rules you make them **persistent** (so that when the machine is rebooted these changes are not lost).

Solution

1) Set Up Port Redirection Rule:

```
sudo iptables -t nat -A PREROUTING -p tcp -s 10.5.5.0/24 --dport 81 -
j DNAT --to-destination 192.168.5.2:80
```

2) Add the masquerading rule:

```
sudo iptables -t nat -A POSTROUTING -s 10.5.5.0/24 -j MASQUERADE
```

3) To make iptables rules persistent:

```
sudo apt install iptables-persistent
```

==========

6

Task

In `/home/bob/certs/` directory you will find 4 files. That's because we generated two self-signed TLS certificates for you. **Delete the 2 files containing the private keys**. But **preserve the certificate files**.

At this point you're left with 2 files containing 2 separate certificates. They both use the RSA algorithm. But one is using 2048 bits for its cryptography purposes, while the other is using 4096. **Delete the certificate that is using 2048 bits**.

Solution

1. To find out which files are not a certificate, run the below command, and if you get output as `Unable to load certificate`, then it is a key.

```
openssl x509 -in file* -noout -text
```

2. To find out the certificate that is using the 2048 bit, run the below command:

```
openssl x509 -in file* -noout -text | grep "Public-Key"
```

==========

7

Task

There is a file at `/opt/aclfile`. Currently no one has permissions to read, write, or execute this file, not even root. But instead of working with regular permissions, use **ACL** for this task.

Add the following to the access control list:

The user called `janet` should be able to **read and write** to `/opt/aclfile`. Just read and write, no execute permission for this **ACL entry**.

Solution

1) Set read and write permissions for janet on /opt/aclfile:

```
sudo setfacl -m u:janet:rw /opt/aclfile
```

==========

8

Task

Add **two security limits** to the configuration of this system:

1. The user called `janet` should have a `hard`` limit so that she can't open more than100processes`.
2. The group called `mail` should have a `soft` limit so users in the group not be able to create files larger than `8192 kilobytes`.

Solution

To enforce security limits edit the security limits configuration file `/etc/security/limits.conf` as below:

```
janet hard nproc 100
@mail soft fsize 8192
```

==========

9

Task

In your home directory you will find a subdirectory called `project`. Navigate to it and then do the following:

1. Add `file1` to the staging area to prepare it for a future commit.
2. Commit this file with the exact following message: `Created first required file`.
3. Now upload your changes to the remote repository already associated with your local repository. Everything is already set up for you. Use the remote repository aliased as origin and upload the `master` branch and password to push the code is `C0ntr0lplan3Pa$$wd`.

Solution

1) Move into the `/home/bob/project` directory
2) Run below command to create a file:

```
touch file1
```

3) Add the file1 to staging by using `git add .` command
4) Commit the changes with the message `Created first required file`:

```
git commit -m "Created first required file"
```

5) Push changes to the `master` branch on remote repo

```
git push origin master
```

==========

10

Task

Perform the following tasks related to **SELinux** on `node01`:

1. First, check if SELinux is running in **enforcing**, **permissive**, or **disabled** mode. Create the file `/opt/selinuxmode.txt`. And write your answer to that file. Just one line where you write a single word: enforcing, permissive, or disabled, according to the status you found.
2. There is a file that has the **wrong SELinux type label**. Please correct that and restore the **default SELinux label** for the file at `/usr/bin/less`.

Credentials to access node01:
Name: `bob`
Password: `caleston123`

Solution

1) Check the current SELinux status on `node01` and store it in file `/opt/selinuxmode.txt`:

```
getenforce | sudo tee /opt/selinuxmode.txt
```

2) Correct SELinux Type Label for `/usr/bin/less`:

```
sudo restorecon -v /usr/bin/less
```

==========

11

Task

Nginx is installed but it's **not running** on `caleston-lp10`. Make the necessary changes so that:

1. Nginx is started immediately.
2. And also, Nginx will start up automatically every time the system boots up.

After starting Nginx, it has spawned at least **3** processes. We are not interested in the **master** process, only the **worker** processes. Under what **username** are

these **worker** processes running? Create a new file at `/opt/nginxuser.txt` and add a single line to it with that username, other than **bob**.

Solution

1) Start Nginx Immediately:
`sudo systemctl start nginx`
2) Enable Nginx to start on boot:
`sudo systemctl enable nginx`
3) List the Nginx processes and store the username at `/opt/nginxuser.txt`:
`ps -ef | grep nginx`
4) Copy the username of worker processes of nginx and store it in `/opt/nginxuser.txt`

==========

12

Task

A basic **LVM** structure exists on the `node01`. Make some changes to it:
The volume group called `volume1` currently only includes `/dev/vdb`.
Add `/dev/vdc` to `volume1`.
We have a logical volume called lv1`. Resize this logical volume to`2GB`.
Credentials to access node01:
Name: `bob`
Password: `caleston123`

Solution

1) Add `/dev/vdc` to `volume1`:
`sudo vgextend volume1 /dev/vdc`
2) Resize the Logical Volume `lv1` to `2GB`:
`sudo lvresize --size 2G /dev/volume1/lv1`

==========

13

Task

Perform the following tasks on `node01`:

1. Install **Git**.
2. In your home directory, download the entire repository from `https://github.com/htop-dev/htop`. By default, this action should create a new subdirectory called **htop** where all the project's files are located.
3. Switch to the **htop** subdirectory. Follow the project's instructions to **build** (compile) the htop application.
4. Install the newly built htop application. By default, the application should be installed in `/usr/local`, in the `bin` subdirectory

Credentials to access node01:
Name: `bob`
Password: `caleston123`
Switch back to `caleston-lp10` once the task is done

Solution

1) Install Git on the `node01` server:
`sudo apt install -y git`
2) Clone the `htop` repository into your home directory and navigate into the local repo:

```
git clone https://github.com/htop-dev/htop
cd htop
```
3) Install the dependencies of `htop`:
```
sudo apt update
sudo apt install libncursesw5-dev autotools-dev autoconf automake
build-essential
```
4) **Install htop** by running the following commands:
```
./autogen.sh
./configure
make
sudo make install
```

==========

14

Task

Create a **virtual machine** with the following parameters on `caleston-lp10`:

1. For the **operating system information parameter** use the `ubuntu22.04`.
2. Name the virtual machine `mockexam2`.
3. Assign `1024 MB` of RAM to this machine.
4. Assign it `one virtual CPU`.
5. Import the disk image `/var/lib/libvirt/images/ubuntu.img` to this virtual machine.
6. At the end of your command, you can add the parameter `--noautoconsole` to avoid waiting for this virtual machine to boot up, and not get stuck in the virtual console after it initializes.

After you create this virtual machine, run a separate command to make `mockexam2` automatically start up every time the system boots up.

Solution

Run the below command to launch vm with `ubuntu22.04 image`:
```
virt-install \
--name mockexam2 \
--ram 1024 \
--vcpus=1 \
--os-variant=ubuntu22.04 \
--import \
--disk path=/var/lib/libvirt/images/ubuntu.img \
--noautoconsole
```
Configure the virtual machine to start automatically on boot:
```
virsh autostart mockexam2
```

==========

15

Task

Create a network bridge between these network interfaces: `[ "eth1", and "eth2"]` on `node01`. Call the bridge `bridge1`. Turn **DHCP off** for IPv4 **for both interfaces**. However, for the **bridge** itself, turn **DHCP** on for **IPv4**.

Credentials to access node01:
Name: bob
Password: caleston123

Solution

To create a bridge network with eth1 and eth2, create a file /etc/netplan/99-bridge.yaml and edit the file as below:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      dhcp4: no
    eth2:
      dhcp4: no
  bridges:
    bridge1:
      dhcp4: yes
      interfaces:
        - eth1
        - eth2
```

Apply the new network configuration using the netplan apply command

==========

16

Task

Perform the following tasks on node01:

1. Remove the Docker image called nginx.
2. Start a container based on the httpd image. **Name** it apache_container. Instruct Docker to redirect connections coming to port 80 on the host to port 80 of this container. Also instruct Docker to restart this container only on failure, with a **maximum number of retries** set to 3 (you will have to look through the correct manual to find the parameter you need).

Credentials to access node01:
Name: bob
Password: caleston123
Switch back to caleston-lp10 once the task is done

Solution

1) Delete the containers using nginx image:

```
docker rm nginx_container
```

2) Delete the nginx image:

```
docker rmi nginx
```

3) Run the below command to create httpd container:

```
docker run --detach --publish 80:80 --name=apache_container --restart=on-failure:3 httpd
```

==========

17

Task

The `node01` was configured to use `LDAP` entries from a certain server. However, some configuration options are wrong. Edit the correct configuration files and fix the following mistakes:

1. Our name service local daemon is configured to look for an **LDAP server** at the **wrong IP address** (currently **10.9.9.8**). Fix this and configure the correct IP which is: **`192.168.121.167`**. Make sure your changes become active after you edit the configuration file.
2. Our system is currently configured to get **group data** and **password data** from the LDAP server but not user data. Configure it to get **user data** as well.

Credentials to access node01:
Name: `bob`
Password: `caleston123`
Switch back to `caleston-lp10` once the task is done

Solution

1. Edit `/etc/nslcd.conf` to change the ldap server:

```
# The user and group nslcd should run as.
uid nslcd
gid nslcd

# The location at which the LDAP server(s) should be reachable.
uri ldap://192.168.121.167/ #updated

# The search base that will be used for all queries.
base dc=example,dc=org

# The LDAP protocol version to use.
#ldap_version 3

# The DN to bind with for normal lookups.
#binddn cn=annonymous,dc=example,dc=net
#bindpw secret
```

2. Edit `/etc/nsswitch.conf` to get user data:

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages
installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         files systemd ldap #updated
group:          files systemd ldap
shadow:         files ldap
gshadow:        files
```

3. Restart the nslcd service:

```
sudo systemctl restart nslcd
```