EVALUATION

A. Questions

1. How does a virtual machine work and what is its purpose?

A virtual machine (VM) is like a computer within a computer. It's a piece of software that mimics a physical computer, creating a separate computer system within one physical hardware system.

Here's how it works:

- You have a physical computer (the host), which has its own operating system and hardware.
- You install a special software called a hypervisor on this host. The hypervisor's job is to create and manage virtual machines.
    - A hypervisor, also known as a virtual machine monitor (VMM), is a piece of software that allows a computer to maintain and operate multiple virtual machines. These virtual machines can run different operating systems and applications on the same physical hardware.
      There are two types of hypervisors:
        - Type 1 (Bare Metal): These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. Examples include Microsoft Hyper-V, VMware ESXi, and Xen.
        - Type 2 (Hosted): These hypervisors run on a conventional operating system just as other computer programs do. A guest operating system runs as a process on the host. VMware Workstation and Oracle VirtualBox are examples of Type 2 hypervisors.

      The hypervisor's role is to allow multiple operating systems to share the same hardware resources, ensuring that each has access to needed resources and that one operating system does not interfere with another. It also provides isolation, ensuring that each virtual machine operates independently of others and as if it were a standalone computer.
- You create a virtual machine using the hypervisor. This virtual machine gets its own portion of the host's resources (like CPU, memory, and storage).
- Inside this virtual machine, you can install an operating system and run applications, just like you would on a physical computer.

The purpose of a VM is to:

- Allow you to run multiple different operating systems on one physical computer.
- Provide a safe, isolated environment to test new software or to run risky operations.
- Make better use of your physical computer's resources by sharing them among multiple virtual machines.
- Easily replicate a specific software environment for testing or distribution.

2. The basic differences between Rocky and Debian

Rocky Linux and Debian are both versions of the Linux operating system, but they're a bit different. Here's how:

Origins and Purpose:

- Debian: Debian has been around for a long time, since 1993. It's known for being very stable and reliable. It's also the foundation for many other versions of Linux, including one called Ubuntu.
- Rocky Linux: Rocky Linux is pretty new, it was first released in 2021. It was created because of changes to another version of Linux called CentOS. Rocky Linux's goal is to be a stable,

ready-to-use version similar to Red Hat Enterprise Linux, which is a popular version used in businesses.

How They Handle Software/ Package Management:

- Debian: Debian uses a system called APT (Advanced Package Tool) to manage software packages. These packages end in .deb.
- Rocky Linux: Rocky Linux uses a system called YUM (Yellowdog Updater, Modified) to manage software packages. These packages end in .rpm.

How Often They Update/ Release Cycle:

- Debian: Debian doesn't update as often because it focuses on being stable and reliable rather than having the latest features.

Remember, both are good choices depending on what you need. Debian is known for being very stable and is a good choice if you want something reliable. Rocky Linux is a good choice if you're looking for something similar to Red Hat Enterprise Linux.

3. Choice of operating system Debian

If you're new to system administration, Debian can be a great choice for the born2beroot project for several reasons:

- Ease of Use: Debian is known for its straightforward and user-friendly approach, which can be a big help when you're just starting out with system administration.
- Strong Community: Debian has a large and active community. This means you can find a lot of tutorials, guides, and forums online to help you if you get stuck.
- Stability: Debian is very stable, which means it runs smoothly and reliably. This can make your life easier when you're learning new skills.
- Package Management: Debian uses the APT package management system, which is easy to use and helps you manage software on your system. This is a key part of system administration.
- Security: Debian is secure and has a dedicated team that quickly releases updates to fix any security issues. This is important when you're learning about system administration, as security is a key part of the job.

Remember, the best choice for you depends on your personal comfort level and learning style. Debian is a good option for many beginners, but you might find another system that suits you better.

4. The difference between aptitude, apt and what APPArmor is

Aptitude vs Apt:

Both Aptitude and Apt are front-end interfaces to dpkg, the package management system used by Debian and its derivatives. They handle dependencies and package installations. Here are some differences:

Purpose:

- apt: Primarily used for handling package management tasks, such as installing, updating, and removing packages on Debian and Ubuntu systems.

- aptitude: A higher-level package management tool that not only manages packages but also provides a text-based interface for handling complex package dependencies and resolutions. It includes features for interactive package management.

User Interface:
- apt: Command-line tool with a straightforward interface.
- aptitude: Text-based user interface, allowing users to interactively resolve package conflicts and dependencies.

Handling Dependencies:
- apt: Generally relies on automatic resolution of dependencies.
- aptitude: Offers more advanced dependency resolution capabilities, allowing users to interactively choose solutions to conflicts.

Package Selection:
- apt: Primarily used for installing, removing, and upgrading packages.
- aptitude: Can also be used for package management, but its interactive interface makes it suitable for more complex scenarios.

Logistical Differences:
- apt: Lightweight and efficient for typical package management tasks.
- aptitude: May be considered a more comprehensive tool due to its interactive nature, but it might have a slightly higher learning curve.

SELinux and AppArmor:

Purpose:
- SELinux (Security-Enhanced Linux): A security module implemented in the Linux kernel to provide enhanced access controls and mandatory access controls (MAC) to strengthen the security of the system.
- AppArmor: A Linux security module that focuses on restricting programs' capabilities rather than using a more complex MAC approach.

Access Control Model:
- SELinux: Implements a strict MAC system where every process and resource has a security context, and access is granted or denied based on policies.
- AppArmor: Uses a profile-based approach, defining what resources an application is allowed to access. It's a more straightforward approach compared to SELinux.

Configuration:
- SELinux: Typically configured through policy rules that define the access permissions for different types of processes and resources.
- AppArmor: Configured through profiles that specify allowed and denied actions for specific applications.

Learning Curve:
- SELinux: Known for having a steeper learning curve due to its complexity and fine-grained control.
- AppArmor: Considered more user-friendly and easier to set up and manage.

Adoption:

- SELinux: Originally developed by the NSA and widely used, especially in enterprise environments.
- AppArmor: Developed by Novell (now part of SUSE) and used by some distributions, including Ubuntu.

Both SELinux and AppArmor aim to improve system security by restricting the actions of processes, but they differ in their approach, complexity, and ease of use. The choice between them often depends on the specific use case and user preferences.

4.1 How they look like

Aptitude and Apt are command-line tools, so they don't have a graphical interface like a typical desktop application. Instead, you interact with them through text commands in a terminal window.

Here's an example of what using Apt might look like:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install packageName
```

Aptitude can also be used in a similar way:

```
$ sudo aptitude update
$ sudo aptitude upgrade
$ sudo aptitude install packageName
```

However, Aptitude also has a text-based user interface if you run it without any commands:

```
$ sudo aptitude
```

This will open a screen in your terminal with a list of packages and options to manage them.

AppArmor is a security module for the Linux kernel, so it doesn't have a user interface. You interact with it through configuration files and command-line tools. For example, you might edit a profile in /etc/apparmor.d/ to configure AppArmor's behavior for a specific program, or use the aa-status command to check the status of AppArmor:

```
$ sudo aa-status
```

This will display information about AppArmor's operation and the profiles currently loaded into the system.

4.2 Package Management

Package management is a method for installing, upgrading, configuring, and removing software packages in a consistent manner. A package is a bundle of files that make up a piece of software.

A package management system, like APT in Debian-based systems or YUM in Red Hat-based systems, keeps track of all the software on your computer, and all the other pieces of software each piece of software needs to work properly (these are called dependencies).

Here's what a package manager does:

- Installing Software: It downloads packages from software repositories, handles all the dependencies (other packages that this package needs to work), and installs them on your system.
- Upgrading Software: It checks the repositories for newer versions of your installed packages, downloads them, and upgrades your system.
- Removing Software: It removes packages and their unused dependencies from your system.
- Resolving Dependencies: It automatically handles dependencies, ensuring that all the required packages are installed.
- Organizing Software: It keeps track of what software is installed, what version it is, and where it's installed.

In short, package management systems simplify the process of managing software on your system, making it easier to keep your software up-to-date and organized.

4.2 The name of package

The name of the package is the identifier used by the package management system (like APT) to refer to a specific piece of software. For example, if you wanted to install the text editor "nano", the package name would be "nano".

Here are some examples of package names:

- nano: A simple, easy-to-use text editor.
- git: A distributed version control system.
- python3: The Python programming language (version 3).
- nginx: A popular web server software.
- mysql-server: The MySQL database server.

You can find the package name for a piece of software in the Debian package repositories, or by searching online. Remember, package names can sometimes vary between different Linux distributions, even if they use the same package management system.

5. What is LVM?

LVM stands for Logical Volume Management. It's a method of storing data on hard drives that makes it easier to manage and more flexible than traditional partitioning methods.

Here's a simple way to understand it:

Imagine you have several baskets (hard drives), and you want to organize your apples (data).

In a traditional partitioning system, you would decide how many apples go into each basket from the start. If one basket gets full, you can't add more apples to it, even if the other baskets are empty.

With LVM, you can create a virtual basket (logical volume) that spans across multiple physical baskets (physical volumes). You can add or remove apples from this virtual basket as needed, and it doesn't matter which physical basket the apples actually go into.

If one physical basket gets full, LVM will automatically put the new apples into the other baskets. If you get a new basket, you can add it to the virtual basket without rearranging all the apples.

In other words, LVM gives you more flexibility to manage your storage space. You can resize, add, or remove logical volumes without worrying about the physical layout of the data.

6. Password Rules

For the password rules, we use the password quality checking library and there are two files the common-password file which sets the rules like upper and lower case characters, duplicate characters etc and the login.defs file which stores the password expiration rules (30 days etc). Sudo nano /etc/login.defs Sudo nano /etc/pam.d/common-password

Check password policy rules

Password expiry: line 160 and 161.

```
vi /etc/login.defs
```

Password policy: line 25.

```
vi /etc/pam.d/common-password
```

6.1 Explain advantages of password policy and advantages and disadvantages of policy implementation

Advantages of Password Policy:

- Better Protection: A good password policy makes it harder for bad guys to break into your stuff. It's like having a stronger lock on your door.
- Meeting Rules: Some businesses have to follow certain rules about security. A password policy helps make sure you're doing what you're supposed to.
- Everyone Knows What to Do: A password policy means everyone follows the same rules. This makes things simpler and more organized.

Advantages of Putting a Policy into Action:

- Being in Charge: When you put a policy into action, you're deciding how things should be done. This helps keep things under control.
- Saving Time and Effort: Policies give clear instructions, which can make things run more smoothly and avoid confusion.
- Avoiding Problems: Policies can help prevent things from going wrong by setting out what's allowed and what's not.

Disadvantages of Putting a Policy into Action:

- People Might Not Like It: Sometimes people don't like change, especially if it makes things harder for them. They might not be happy about a new policy.
- It Can Cost Money: There might be costs involved in putting a new policy into action, like training people or checking that the rules are being followed.
- It Can Be Too Rigid: Sometimes policies can be too strict and not allow for flexibility. This can make it hard to deal with unusual situations or to try out new ideas.

7. UFW (Uncomplicated Firewall)

UFW is a interface to modify the firewall of the device without compromising security. You use it to configure which ports to allow connections to and which ports to close. This is useful in conjunction with SSH, can set a specific port for it to work with.

8. What is SSH?

SSH is a cryptographic network protocol used to securely access and manage network devices and servers over an unsecured network. It provides a secure channel over an insecure network, typically using a pair of public and private keys for authentication.

What is SSH?

Definition: SSH, or Secure Shell, is a way to securely connect to another computer over a network, typically the internet.

How Does SSH Work?

- Secure Communication: SSH provides a secure way to communicate between computers. It encrypts the data exchanged during the communication, making it difficult for others to intercept and understand.
- Authentication: SSH uses a system of keys to ensure that the right person or system is connecting. It's like having a digital ID card.
- Key Pair: SSH works with a pair of keys - a public key and a private key. The public key is shared with others, while the private key is kept secret.
- Key Exchange: When you connect to a remote system using SSH, your computer and the remote system exchange keys. This happens at the beginning of the connection and ensures that both systems are who they say they are.
- Encryption: Once the keys are exchanged, all the data sent between your computer and the remote system is encrypted. This means even if someone intercepts the communication, they can't understand it without the right keys.

Why Use SSH?

- Secure Access: SSH provides a secure way to access and manage remote systems. It's commonly used by administrators and developers to connect to servers.
- Encrypted Data: The data exchanged during an SSH connection is encrypted, adding an extra layer of security. This is crucial when working with sensitive information.
- Remote Control: With SSH, you can control another computer as if you were physically present at that machine. This is especially useful for managing servers.

Common Use Cases:

- Remote Server Management: Administrators use SSH to manage servers remotely, performing tasks like installing software, checking logs, and configuring settings.
- Secure File Transfer: SSH is used for secure file transfers between computers. The scp (secure copy) command is commonly used for this purpose.
- Tunneling and Port Forwarding: SSH can create secure tunnels for transferring data or accessing services on a remote machine. This is helpful for securely accessing resources behind a firewall.

In a nutshell, SSH is like a secure tunnel that allows you to communicate with another computer in a way that keeps your information private and ensures that you are connecting to the right place. It's a fundamental tool for secure remote access and data transfer.

9. What is Cron?

Cron or cron job is a command line utility to schedule commands or scripts to happen at specific intervals or a specific time each day. Useful if you want to set your server to restart at a specific time each day.

`cd /usr/local/bin` – to show monitoring.sh
`sudo crontab -u root -e` – to edit the cron job
`change script to */1 * * * * sleep 30s && script path` – to run it every 30 seconds, delete the line to stop the job from running.

B. CHECK

B.1. SIMPLE SETUP

- ☐ Script running every 10min
- ☐ No graphical user interface
- ☐ Password requested on boot up
- ☐ Login with lkilpela as a created user (which is not a root)
- ☐ Password must follow rules (2 days min, 7, 30 days max)

```
Sudo chage -l username
```

- ☐ Evaluator checks that the UFW service is started

```
sudo ufw status //look for status: active
```

- ☐ Evaluator checks that the SSH service is started

```
sudo systemctl status ssh
Or
sudo service ssh status
```

- ☐ Evaluator checks that the chosen operating system is Debian

```
cat /etc/os-release | grep PRETTY_NAME
Or
Lsh_release -a || cat /etc/os-release
```

B.2. USER

- ☐ Check a user with the login of the student being evaluated is present on the VM & user belongs to “sudo” and “user42” groups

```
gettend group sudo
getend group user42
```

B.3. PASSWORD POLICY CHECK

- ☐ Check that lkilpela is member of sudo and user42 groups

```
groups lkilpela
```

- [ ] Check password policy rules

Password expiry: line 160 and 161.

```
vi /etc/login.defs
```

Password policy: line 25.

```
vi /etc/pam.d/common-password
```

- [ ] Create a new user

```
sudo adduser new_username
```

- [ ] Assign password

```
#Confirm it follows the password policy
getent group sudo
```

- [ ] Explain how password rules were set up

```
vi /etc/pam.d/common-password
```

Password rules are set up using PAM (Pluggable Authentication Modules) and the pam_pwquality.so module. Here's a step-by-step guide on how it's done:

1. Install the necessary package: *First, you need to install the `libpam-pwquality` package which provides the `pam_pwquality.so` module. You can do this with the following command:*

```
sudo apt-get install libpam-pwquality
```

2. Configure the module: *Next, you need to configure the module to enforce your password rules. This is done by editing the `/etc/pam.d/common-password` file. You can open this file with a text editor like `nano`:*

```
sudo nano /etc/pam.d/common-password
```

In this file, you'll find a line that looks something like this:

```
password    requisite    pam_pwquality.so retry=3
```

You can add your password rules to the end of this line. For example, to enforce a minimum length of 10 characters, you would change it to:

```
password    requisite    pam_pwquality.so retry=3 minlen=10
```

You can add multiple rules to this line. For example, to also require at least one digit and one uppercase letter, you would use:

```
password    requisite    pam_pwquality.so retry=3 minlen=10 dcredit=-1 ucredit=-1
```

Here, `dcredit=-1` means at least one digit is required, and `ucredit=-1` means at least one uppercase letter is required.

3. Save and exit: *After you've added your rules, save the file and exit the text editor. The new rules will take effect immediately.*

- [ ] Create group *evaluating* and add created user

```
sudo groupadd evaluating
sudo groupadd new_user evaluating or sudo usermod -aG evaluating your_new_username
```

- ☐ Check that user belongs to new group

```
groups new_user
Or
getent group evaluating
```

- ☐ Explain advantages of password policy

Advantages of Password Policy:

- *Better Protection: A good password policy makes it harder for bad guys to break into your stuff. It's like having a stronger lock on your door.*
- *Meeting Rules: Some businesses have to follow certain rules about security. A password policy helps make sure you're doing what you're supposed to.*
- *Everyone Knows What to Do: A password policy means everyone follows the same rules. This makes things simpler and more organized.*

- ☐ Explain advantages and disadvantages of policy implementation

Advantages of Putting a Policy into Action:

- *Being in Charge: When you put a policy into action, you're deciding how things should be done. This helps keep things under control.*
- *Saving Time and Effort: Policies give clear instructions, which can make things run more smoothly and avoid confusion.*
- *Avoiding Problems: Policies can help prevent things from going wrong by setting out what's allowed and what's not.*

Disadvantages of Putting a Policy into Action:

- *People Might Not Like It: Sometimes people don't like change, especially if it makes things harder for them. They might not be happy about a new policy.*
- *It Can Cost Money: There might be costs involved in putting a new policy into action, like training people or checking that the rules are being followed.*
- *It Can Be Too Rigid: Sometimes policies can be too strict and not allow for flexibility. This can make it hard to deal with unusual situations or to try out new ideas.*

B4. HOSTNAME & PARTITIONS

- ☐ Check that the hostname of the machine is lkilpela42 (login of the evaluated)

```
uname -n
# or
hostnamectl
```

- ☐ Modify hostname with evaluator login and restart VM to confirm change

```
sudo adduser new_user sudo
sudo login new_user
```

```
sudo vi /etc/hostname # change to new_user42
sudo reboot
# Or
sudo hostnamectl set-hostname new_hostname
sudo reboot
```

- ☐ Restore original hostname and restart VM to confirm change

```
sudo vi /etc/hostname # change to lkilpela42
sudo reboot
# Or
sudo hostnamectl set-hostname new_hostname
sudo reboot
```

- ☐ Ask the evaluated how to view the partitions for the VM

```
lsblk
```

- ☐ Compare partition output with example in subject

```
NAME                    MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                       8:0    0    8G  0 disk
|-sda1                    8:1    0  476M  0 part  /boot
|-sda2                    8:2    0    1K  0 part
`-sda5                    8:5    0  7.5G  0 part
  `-sda5_crypt          254:0    0  7.5G  0 crypt
    |-LVMGroup-root     254:1    0  1.9G  0 lvm   /
    |-LVMGroup-swap     254:2    0  952M  0 lvm   [SWAP]
    |-LVMGroup-home     254:3    0  952M  0 lvm   /home
    |-LVMGroup-var      254:4    0  952M  0 lvm   /var
    |-LVMGroup-srv      254:5    0  952M  0 lvm   /srv
    |-LVMGroup-tmp      254:6    0  952M  0 lvm   /tmp
    `-LVMGroup-var--log 254:7    0    1G  0 lvm   /var/log
sr0                      11:0    1 1024M  0 rom
```

- ☐ Brief explanation of LVM & how LVM works

LVM in a Nutshell:
1. Abstraction Layer: LVM acts like a smart organizer for your computer's storage. It creates a layer of abstraction, making it easier to manage your hard drives.
2. Building Blocks: LVM uses three main building blocks: Physical Volumes (PVs), Volume Groups (VGs), and Logical Volumes (LVs).
3. Physical Volumes (PVs): Think of PVs as your actual hard drives or SSDs. LVM groups these physical devices together.
4. Volume Groups (VGs): Volume Groups are like virtual containers that hold one or more physical drives. This grouping allows for more efficient use of storage.
5. Logical Volumes (LVs): LVs are like virtual partitions created within Volume Groups. Your operating system interacts with these logical partitions, treating them like regular hard drive partitions.
6. Dynamic Resizing: LVM allows you to resize your Logical Volumes on-the-fly. It's like adjusting the size of your virtual partitions without turning off your computer.
7. Flexibility: LVM offers flexibility. If you need more space, you can add a new hard drive to the Volume Group, and LVM takes care of distributing the data.
8. Snapshots: LVM can take snapshots, which are like instant backups of your data at a specific point in time. Useful for testing or making backups without stopping everything.

How LVM Works:

- *Grouping Drives: LVM starts by grouping your physical drives (PVs) into a Volume Group (VG).*
- *Creating Logical Partitions: Within the Volume Group, logical partitions (LVs) are created. These act like regular hard drive partitions but with more flexibility.*
- *Interacting with LVs: Your operating system interacts with these logical partitions, thinking they are regular partitions, but LVM is managing things behind the scenes.*
- *Dynamic Changes: Need more space? LVM allows you to add a new hard drive to the Volume Group or resize existing partitions without shutting down your computer.*
- *Snapshots for Safety: LVM can create snapshots, providing a quick and safe way to back up your data or test changes without risking your main system.*

Why LVM?

- *Flexibility & Easy Maintenance: LVM makes it easy to manage and expand your storage without disrupting your system.*
- *Efficient Use of Space: It efficiently uses space by pooling multiple drives together.*
- *Snapshots for Safety: Snapshots provide a safety net for backups or testing changes without affecting your main data.*

In essence, LVM is like a helpful organizer for your computer's storage, making it more flexible, adaptable, and easy to manage.

B5. SUDO

☐ Check sudo program is properly installed

```
dpkg -l | grep sudo
```

☐ Assign new user to sudo group

```
sudo adduser new_user sudo
```

☐ Explain value and operation of sudo using examples

```
sudo visudo ls
```

sudo is a command in Linux, it stands for "SuperUser DO" that lets you run commands as a different user, typically the superuser (root).

Value of sudo:

- *Security: It's safer to use sudo for commands that need root privileges than to log in as root.*
- *Auditability: sudo logs all commands, which helps track and review actions.*
- *Control: Administrators can control who can use sudo and what they can do with it.*

Operation of sudo:

For example, to update your system's packages, you'd use sudo because this requires root privileges:

```
apt-get update # Error 13: Permission denied
sudo apt-get update
```

To edit a system file like /etc/hosts, you'd also need sudo:

```
sudo nano /etc/hosts
```

Remember, sudo is powerful and should be used carefully to avoid damaging your system.

- ☐ Show the implementation of sudo rules is primarily defined in the /etc/sudoers file (read - only). If want to change the sudo rules then use sudo visudo

```
sudo nano /etc/sudoers
```

- ☐ Verify that the /var/log/sudo/ folder exists and has a file sudo.log, folder 00 & seq file

```
sudo ls -la /var/log/sudo/
```

- ☐ Check contents of files in this folder

```
sudo cat ls /var/log/sudo/sudo.log
```

- ☐ Check there is a history of commands using sudo

```
sudo cat /.../log # Input log
sudo cat /.../ttyout # Output log
```

- ☐ Run a command using sudo and check if files updated

```
sudo apt update
sudo ls /var/log/sudo/seq # check the seq number changed and one folder is added
```

B5. UFW/ Uncomplicated Firewall

- ☐ Check that UFW is properly installed

```
dpkg -l | grep ufw
# Or
sudo ufw status numbered
```

- ☐ Check that it is working properly

```
sudo ufw status
```

- ☐ Explain what UFW is and the value of using it

UFW, or Uncomplicated Firewall, is a tool for managing firewall rules in a Linux system. It's designed to be user-friendly with simple commands, making it easier to manage network traffic rules.

Value of UFW:

- *Ease of use: UFW simplifies the process of managing a firewall on a Linux system. It provides a user-friendly interface to the more complex iptables command, making it easier to allow or deny network connections.*

  **iptables = like a security guard for your computer. It's a tool in Linux that helps control incoming and outgoing network traffic. It sets up rules for who can access your computer and how, helping to keep it safe from unauthorized access.*

  Simple example of using iptables to block all incoming traffic to your computer except for SSH connections (which use port 22):

  ```
  # First, block all incoming traffic
  sudo iptables -P INPUT DROP

  # Then, allow SSH connections
  sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
  ```

  In this example:

- *The first command sets the default policy for incoming traffic to DROP, which means all incoming traffic will be blocked.*
- *The second command adds a rule (-A stands for "append") to the INPUT chain. This rule matches TCP traffic (-p tcp) destined for port 22 (--dport 22), which is the standard port for SSH. The -j ACCEPT part tells iptables to allow this traffic.*

Remember, iptables rules can be complex and powerful, and incorrect rules can disrupt your network connectivity. Always be careful when modifying iptables rules.

- *Security: By controlling which services and connections can send or receive traffic to your system, UFW helps to secure your system against unauthorized access or attacks.*

For example, to allow traffic on a specific port (like 80 for a web server), you would use the command:

```
sudo ufw allow 80
```

And to deny traffic on a specific port (like 25, often used for sending mail), you would use the command:

```
sudo ufw deny 25
```

In short, UFW provides a simpler way to manage your firewall, helping to secure your system with less effort.

☐ List active rules should include one for port 4242

```
sudo ufw status | grep 4242
```

☐ Add a new rule for port 8080

```
sudo ufw allow 8080
sudo ufw status
```

☐ Delete the new rule

List rules numbered

```
sudo ufw status numbered
```

Delete rule

```
sudo ufw delete $NUMBER
```

B6. SSH

☐ Check that the SSH service is properly installed

```
dpkg -l | grep openssh-server
```

☐ Check that it is working properly

```
sudo service ssh status // check if it actives and port 4242
```

☐ Explain what SSH is and the value of using it

SSH, or Secure Shell, is like a secret passageway that lets you safely access and control another computer from your own computer, even if they're miles apart.

Value of SSH:

Safety: It keeps your connection private, so no one can spy on what you're doing.
Control: It lets you run commands on the other computer as if you were sitting right in front of it.
So, if you need to access a computer in another location (like a web server), SSH is a safe and powerful way to do it.

☐ Verify that the SSH service only uses port 4242

```
sudo service ssh status | grep listening
# or check configs
sudo vi /etc/ssh/sshd_config
sudo vi /etc/ssh/ssh_config
```

☐ Login with SSH with newly created user from host machine

```
ssh lkilpela@127.0.0.1 -p 4242 # or
ssh lkilpela@0.0.0.0 -p 4242 # or
ssh lkilpela@localhost -p 4242
```

☐ Make sure you cannot use SSH login with root user

```
lkilpela@lkilpela42:~$ login root
login: Cannot possibly work without effective root
```

B7. SCRIPT MONITORING

Script inputted in the monitoring.sh file to display system information

```
cd/usr/local/bin && nano monitoring.sh
*display script: lkilpela@lkilpela42:/usr/local/bin$ ./monitoring.sh
```

```
Broadcast message from lkilpela@lkilpela42 (pts/0) (Sun Dec 31 21:58:01 2023):

      #Architecture: Linux lkilpela42 6.1.0-16-amd64 #1 SMP PREEMPT_DYNAMIC Debian
6.1.67-1 (2023-12-12) x86_64 GNU/Linux
      #CPU physical: 1
      #vCPU: 2
      #Memory Usage: 248/1967MB (12.66%)
      #Disk Usage: 1734/7Gb (26%)
      #CPU load: 0.0%
      #Last boot: 2024-01-03 14:46
      #LVM use: yes
      #Connections TCP: 1 ESTABLISHED
      #User log: 2
      #Network: IP 10.0.2.15  (08:00:27:3d:82:e7)
      #Sudo: 29 cmd


#!/bin/bash

# Get system architecture
# `uname -a` provides detailed system information
architecture=$(uname -a)

# Get the number of physical CPUs
# `lscpu` displays CPU architecture information
physical_cpu=$(lscpu | grep 'Socket(s):' | awk '{print $2}')

# Get the number of virtual CPUs
# `nproc` returns the number of processing units available
virtual_cpu=$(nproc)

# Get memory usage
# `free -m` displays the total amount of free and used physical and swap memory in
the system
```

```
memory_usage=$(free -m | awk '/^Mem:/{printf("%s/%sMB, (%.2f%%) \n", $3, $2,
$3/$2*100)}')

# Get disk usage
# `df -Bg` reports file system disk space usage in Gb
disk_usage=$(df -Bg | grep "^/dev" | grep -v "/boot" | awk '{total+=$2} {used+=$3}
END {printf "%d/%dGb (%d%%)\n", used, total, used/total*100}')

# Get CPU load
# `top -bn1` provides a dynamic real-time view of a running system
cpu_load=$(top -bn1 | grep '^%Cpu' | awk '{printf("%.1f%%"), $2 + $4 + $6}')

# Get the last reboot time
# `who -b` shows system boot time
last_reboot=$(who -b | awk '{print $3, $4, $5}'| xargs -I{} date -d {} +"%Y-%m-%d
%H:%M")

# Check if LVM is in use
# `lsblk` lists block devices and `grep -q 'lvm'` checks for lvm
lvm_use=$(lsblk | grep -q 'lvm' && echo "Yes" || echo "No")

# Get the number of established TCP connections
# `netstat -an` prints network connections, routing tables, interface statistics,
etc.
tcp_connections=$(netstat -an | grep ESTABLISHED | wc -l)

# Get the number of logged in users
# `who` shows who is logged on
user_log=$(who | wc -l)

# Get network information
# `hostname -I` gives the IP address of the host and `ip addr` shows network
device configuration
network=$(hostname -I | awk '{printf "%s ", $1}' ; ip addr | grep "link/ether" |
awk '{printf "(%s)", $2}')

# Get the number of sudo commands executed
# `sudo grep 'COMMAND=' /var/log/sudo/sudo.log` finds sudo commands in the log
sudo_commands_count=$(sudo grep 'COMMAND=' /var/log/sudo/sudo.log | wc -l)

# Display the gathered information
# `wall` sends a message to everybody's terminal
wall "#Architecture: $architecture
    #CPU physical: $physical_cpu
    #vCPU: $virtual_cpu
    #Memory Usage: $memory_usage
    #Disk Usage: $disk_usage
    #CPU load: $cpu_load
    #Last boot: $last_reboot
    #LVM use: $lvm_use
    #Connections TCP: $tcp_connections ESTABLISHED
    #User log: $user_log
    #Network: IP $network
    #Sudo: $sudo_commands_count cmd"
```

- ☐ Explanation of the monitoring script by showing the code

Architecture - operating system & its kernel version

```
arc=$(uname -a)
```

uname (short for unix name) is a computer program in Unix and Unix-like computer operating systems.The uname -a command in Linux provides comprehensive information about the system. It prints all system information at once, including the kernel name, hostname, kernel release number, kernel version, machine hardware name (architecture), processor type, hardware platform, and operating system.

```
#Architecture: Linux lkilpela42 6.1.0-16-amd64 #1 SMP PREEMPT_DYNAMIC Debian
6.1.67-1 (2023-12-12) x86_64 GNU/Linux
```

- ***Linux****: The operating system name.*
- ***lkilpela42****: The hostname of the system (this can be set by the user and may vary).*
- ***6.1.0-16-amd64****: The kernel release number. This includes the version and build details of the Linux kernel that's currently being used.*
- ***#1 SMP PREEMPT_DYNAMIC Debian 6.1.67-1 (2023-12-12****): This part provides more details about the kernel. "SMP" stands for Symmetric MultiProcessing, which means the kernel can use multiple CPUs simultaneously. "PREEMPT_DYNAMIC" is a kernel configuration option that allows for better real-time performance. The "Debian 6.1.67-1" is the specific distribution and version of Linux that's being used. The date in parentheses is the build date of this kernel version.*
- ***x86_64****: This is the architecture of the hardware, indicating that it's a 64-bit system.*
- ***GNU/Linux****: This indicates that the system is running a version of Linux that's part of the GNU project.*

Physical processor (CPU = Central Processing Unit)

```
lscpu | grep 'Socket(s):' | awk '{print $2}'
```

** CPU: (Central Processing Unit) is the primary component of a computer that performs most of the processing. It's often referred to as the "brain" of the computer, executing instructions of a software program by performing basic arithmetic, logical, control, and input/output operations.*

In the context of computer hardware, a physical processor refers to a CPU (Central Processing Unit).

However, in terms of terminology used in commands like lscpu, the term "Socket(s)" refers to the number of physical processors or CPUs. Each socket hosts a CPU. So, when you see "Socket(s)" in the output of lscpu, it's referring to the number of physical CPUs in the system. In a multi-core processor, each core is effectively a CPU itself, but they are all housed in a single physical processor or socket. So, to summarize, a socket hosts a physical processor, and a physical processor (CPU) can have one or more cores.

- *lscpu: This command displays information about the CPU architecture.*
- *grep 'Socket(s):': This command filters the output of lscpu to find the line that contains 'Socket(s):', which represents the number of physical CPUs (or sockets).*
- *awk '{print $2}': This command prints the second field on that line, which is the number of sockets.*

Virtual processor (virtual CPU cores)

```
vcpu=$(nproc)
```

nproc: print the number of processing units available to the current process, which is typically the number of CPU cores, including both physical and hyperthreaded cores.

Memory_usage

Percentage of RAM memory usage

```
memory_usage=$(free -m | awk '/^Mem:/{printf("%s/%sMB, (%.2f%%) \n", $3, $2,
$3/$2*100)}')
```

**RAM (Random Access Memory) is a type of computer memory that temporarily stores data that the computer is currently using. It's often referred to as the computer's "short-term memory". The more RAM a computer has, the more data it can process quickly at the same time.*

Calculates the total and used memory (RAM) on your system in megabytes and assigns the result to the variable memory_usage. Here's a breakdown:

- ***free -m:** This command displays the amount of free and used memory in your system. The -m option makes the output display sizes in megabytes.*
- ***awk '/^Mem:/{printf("%s/%sMB, (%.2f%%) \n", $3, $2, $3/$2*100)}'**: This part of the command does the following:*
  - ***/^Mem:/:** Filters the output of free -m to include only the line that starts with 'Mem:', which represents the main memory.*
  - ***printf("%s/%sMB, (%.2f%%) \n", $3, $2, $3/$2*100):** Prints the used memory ($3), the total memory ($2), and the percentage of used memory ($3/$2*100). The %.2f format specifier is used to limit the percentage to two decimal places.*
- ***memory_usage=$()**: This part of the command assigns the result of the operation (the output of the awk command) to the variable memory_usage.*

So, after running this command, memory_usage will hold a string like "used/totalMB, (percentage%)", representing the total and used memory on your system, in megabytes.

Percentage of disk memory usage

```
disk_usage=$(df -Bg | grep "^/dev" | grep -v "/boot" | awk '{total+=$2} {used+=$3}
END {printf "%d/%dGb (%d%%)\n", used, total, used/total*100}')
```

**Disk memory, often referred to as disk storage, is a type of data storage where information is written to and read from a rotating disk. This is typically a hard disk drive (HDD) or a solid-state drive (SSD) in modern computers.*

Calculates the total and used disk space of all mounted filesystems (except for the boot partition) on your system and assigns the result to the variable disk_usage. Here's a breakdown:

- ***df -Bg:** This command displays the amount of disk space used and available on your filesystems. The -Bg option makes the output display sizes in gigabytes.*
- ***grep "^/dev":** This part of the command filters the output of df -Bg to include only lines that start with '/dev/', which represent mounted filesystems.*
- ***grep -v "/boot"**: This part of the command filters the output to exclude lines that contain '/boot', which represents the boot partition. The /boot partition is excluded because it contains static files for system boot that don't typically change in size, thus it doesn't accurately reflect the dynamic usage of disk space for user data and applications.*
- ***awk '{total+=$2} {used+=$3} END {printf "%d/%dGb (%d%%)\n", used, total, used/total*100}'**: This part of the command does the following:*
  - ***{total+=$2}:** Adds up the second field (the total size of the filesystem) on each line.*
  - ***{used+=$3}**: Adds up the third field (the used size of the filesystem) on each line.*
  - ***END {printf "%d/%dGb (%d%%)\n", used, total, used/total*100}**: At the end, prints the total used size, the total size, and the percentage of used size.*
- ***disk_usage=$():** This part of the command assigns the result of the operation (the output of the awk command) to the variable disk_usage.*

So, after running this command, disk_usage will hold a string like "used/totalGb (percentage%)", representing the total and used disk space of all mounted filesystems (except for the boot partition) on your system, in gigabytes.

CPU load

**CPU load, or load average, is a measure of the amount of computational work that a computer system performs. It represents the average number of active processes over a period of time.*

```
cpu_load=$(top -bn1 | grep '^%Cpu' | cut -c 9- | xargs | awk '{printf("%.1f%%"), $1 + $2 + $3}'
```

```
%Cpu(s):  5.0 us,  2.5 sy,  0.0 ni, 92.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

- *%us: User CPU time, the time the CPU has spent running users' processes that are not niced.*
- *%sy: System CPU time, the time the CPU has spent running the kernel and its processes.*
- *%ni: Nice CPU time, the time the CPU has spent running users' niced processes.*
- *%id: Idle CPU time, the time the CPU has spent doing nothing.*
- *%wa: I/O wait time, the time the CPU has spent waiting for I/O operations to complete.*
- *%hi: Hardware IRQ time, the time the CPU has spent servicing hardware interrupts.*
- *%si: Software IRQ time, the time the CPU has spent servicing software interrupts.*
- *%st: Steal time, the time the operating system wanted to execute, but was not allowed to by the virtual machine manager.*

To calculate the CPU load, you typically want to consider the User CPU time (%us), System CPU time (%sy), and possibly Nice CPU time (%ni). These represent the time the CPU has spent doing work. The other values represent idle time or time spent waiting, which are not usually considered part of the CPU load.

- *top -bn1: The top command provides a dynamic real-time view of a running system. The -b option runs top in batch mode, which could be useful for sending output from top to other programs or to a file. The -n1 option updates the top command once and then exits.*
- *grep '^%Cpu': This filters the output of top to only include lines that start with %Cpu, which is where top displays CPU usage.*
- *cut -c 9-: This cuts out the first 8 characters of each line, leaving only the CPU usage percentages.*
- *xargs: This command is used to build and execute commands from standard input. It converts the input into a single line.*
- *awk '{printf("%.1f%%"), $1 + $2 + $3}': This uses awk, a programming language for text processing, to add up the first three numbers on the line (which represent the %us, %sy, and %ni CPU usage percentages reported by top) and print the result with one decimal place.*

Last reboot

```
last_reboot=$(who -b | awk '{print $3, $4, $5}'| xargs -I{} date -d {} +"%Y-%m-%d %H:%M")
```

- *who -b: This command outputs the time of the last system boot.*
- *awk '{print $3, $4, $5}': This command processes the output of who -b and prints the 3rd, 4th, and 5th fields, which correspond to the month, day, and time of the last boot.*
- *xargs -I{} date -d {} +"%Y-%m-%d %H:%M": This command takes the output from awk, which is piped into xargs. The -I{} option allows us to use {} as a placeholder for the incoming piped data. date -d {} interprets the incoming data as a date, and +"%Y-%m-%d %H:%M" formats the date in "Year-Month-Day Hour:Minute" format.*
    - *xargs -I{}: This part of the command reads items from standard input, delimited by blanks or newlines, and executes the command once for each item. The -I{} option allows us to use {} as a placeholder for the incoming items.*
    - *date -d {}: This part of the command interprets the incoming item as a date. The -d option is used to specify the input date string.*
    - *+"%Y-%m-%d %H:%M": This part of the command specifies the output format for the date. It's formatted as "Year-Month-Day Hour:Minute".*
    - *So, if you pipe a date string into this command, it will output the date in the format "Year-Month-Day Hour:Minute". For example:*
      echo "Dec 31 20:04" | xargs -I{} date -d {} +"%Y-%m-%d %H:%M"
      This will output 2020-12-31 20:04 (or whatever the year was for the last December 31st at 20:04).

The result is stored in the last_reboot variable. So, after running this command, last_reboot will hold the date and time of the last system boot in the format YYYY-MM-DD HH:MM.

LVM is active or not

```
lvm_use=$(lsblk | grep -q 'lvm' && echo "Yes" || echo "No")
```

- *lsblk: Lists all block devices on your system.*
- *Grep -q 'lvm': Searches for the string 'lvm' in the output of lsblk. If it finds 'lvm', it means that LVM is used on your system. The -q option makes grep not output anything*
- *&& echo "Yes" || echo "No": If the grep command succeeds (i.e., if 'lvm' is found), it echoes "Yes". Otherwise, it echoes "No".*

After running this command, the lvm_use variable will hold the string "Yes" if LVM is used on your system and "No" otherwise.

tcp_connections

```
tcp_connections=$(netstat -an | grep ESTABLISHED | wc -l)
```

**TCP connections are established, reliable communication links between two devices on a network using the Transmission Control Protocol (TCP). They are used for transmitting data over the internet in a reliable and ordered manner.*

- ***netstat -an**: This command lists all network connections, routing tables, interface statistics, and more.*
  - *-a: This option stands for 'all', which means it will display both listening and non-listening (for TCP this means established) connections.*
  - *The -n option in netstat -an is used to show network addresses as numbers. Without -n, netstat would try to show names instead of numbers, like converting IP addresses to hostnames, which can take more time. So, -n makes the command run faster and shows the output in a straightforward, numeric way.*
- ***grep ESTABLISHED:** This command filters the output of netstat -an to only include established connections.*
- ***wc -l**: This command counts the number of lines in the output of grep ESTABLISHED, which corresponds to the number of established connections.*

After running this command, tcp_connections will hold the number of active TCP connections.

Please note that you might need to run this command with sudo to get the correct output, depending on your system's permissions.

users_logged_in

```
user_log=$(who | wc -l)
```

who: This command lists all the users currently logged into the system.

- *wc -l: This command counts the number of lines in the output. Since who outputs one line per logged-in user, this effectively counts the number of users.*
- *user_log=$(...): This part assigns the output of the command inside the parentheses to the variable user_log.*

So, after running this command, user_log will hold the number of users currently logged into the system.

Network

```
network=$(hostname -I | awk '{printf "%s ", $1}' ; ip addr | grep "link/ether" |
awk '{printf "(%s)", $2}')
```

ipv4_address

```
ip_address=$(hostname -I)
```

ip=$(hostname -I): This command retrieves the IP address of your system. The hostname -I command outputs all network addresses of the host. The output is assigned to the ip variable.

mac_address

```
mac_address=$(ip addr | grep "ether" | awk '{print $2}')
```

- *ip addr: This command displays network configuration, including network interfaces and their respective IP and MAC addresses.*
- *grep "ether": This command filters the output of ip addr to only include lines containing "ether", which are the lines that include the MAC addresses.*
- *awk '{print $2}': This command extracts the second field from these lines, which is the MAC address.*
- *mac=$(...): This part assigns the output of the command inside the parentheses to the variable mac.*

So, after running this command, mac will hold the MAC address of your system. Note that if your system has multiple network interfaces, this command will return the MAC address of one of them.

sudo_commands_count

```
sudo_commands_count=$(sudo grep 'COMMAND=' /var/log/sudo/sudo.log | wc -l)
```

- *sudo grep 'COMMAND=' /var/log/sudo/sudo.log: This command searches for the string 'COMMAND=' in the sudo logs, which are typically located at /var/log/sudo/sudo.log. Each 'COMMAND=' entry corresponds to a command executed with sudo.*
- *wc -l: This command counts the number of lines in the output of grep, which corresponds to the number of sudo commands.*
- *sudo_commands_count=$(...): This part assigns the output of the command inside the parentheses to the variable sudo_commands_count.*

After running this command, sudo_commands_count will hold the number of commands executed with sudo.

☐ What is cron

Cron is like an automatic scheduler for your computer. It's a tool on Linux systems that you can set up to run tasks or commands at specific times.

For example, you could use cron to automatically back up your files every night while you sleep, or to check for updates once a week. You just tell cron what you want it to do and when, and it takes care of the rest.

☐ How to set up the script to run every 10mins

```
sudo crontab -e
```

Add following line

```
*/10 * * * * /home/monitoring.sh
```

☐ Verify correct functioning of the script

Check print out in console.

☐ Change run of script to every minute

```
sudo crontab -e
```

Add following line

```
*/1 * * * * /home/monitoring.sh
```

- [ ] Make the script stop running after reboot without modifying it

Remove the scheduling line on the crontab

```
sudo crontab -e
```

Remove following line/s

```
@reboot /home/monitoring.sh
*/1 * * * * /home/monitoring.sh
```

- [ ] Restart server

```
sudo systemctl stop cron
sudo cronstart
```

- [ ] Check script still exists in the same place
- [ ] Check that its rights have remained the same
- [ ] Check that it has not been modified

```
sudo reboot
sudo crontab -u -root -e
```

IMPLEMENTATION

Part 1. Configurating Your Virtual Machine

**The lsblk command in Linux stands for "list block devices". It displays information about all available or the specified block devices. Block devices include hard drives, flash drives, and other storage devices. The lsblk command lists these devices in a tree-like format by default.*

1.1 Installing Sudo

1. First type su - to login in as the root user.
2. Then type apt update
3. Then type apt upgrade
4. Then type apt install sudo
5. Then type usermod -aG sudo your_username to add user in the sudo group
6. To check if user is in sudo group, type getent group sudo
7. Type sudo visudo to open sudoers file
8. Lastly find - # User privilege specification, type your_username ALL=(ALL) ALL
   root ALL=(ALL:ALL) ALL: *This means that the root user (the main administrator of the system) has permission to execute any command on any host as any user or any group. In simpler terms, the root user can do anything on the system without any restrictions.*

   your_username ALL=(ALL) ALL: *This means that the user with the username your_username has permission to execute any command on any host as any user. In simpler terms, the user your_username can also do anything on the system, but only as themselves, not as any other user or group.*

   These rules are typically defined in the sudoers file, which controls who can run what commands as which users on your system. It's a way to give certain users the ability to perform administrative tasks without giving them full root access.

1.2 Installing Git and Vim

1. Then type apt install git/vim to install Git/VIM
2. Then type git/vim --version to check the Git/VIM Version

1.3 Installing and Configuring SSH (Secure Shell Host)

Install SSH

1. Type sudo apt update
2. Type sudo apt install openssh-server

Start & Enable SSH

3. Type sudo systemctl status ssh to check SSH Server Status
4. Type sudo systemctl start ssh
5. Type sudo systemctl enable ssh

Configure SSH

The main configuration file for SSH is located at /etc/ssh/sshd_config. You can edit this file to change the default SSH settings:

6. Type sudo nano/vim /etc/ssh/sshd_config
7. Find this line #Port22 *(The port that SSH listens on. The default is 22.)*
8. Change the line to Port 4242 without the # (Hash) in front of it

9. To disable SSH login as root irregardless of authentication mechanism, replace below line #PermitRootLogin prohibit-password with PermitRootLogin no
10. Save and Exit Vim/nano
11. Then type sudo grep Port /etc/ssh/sshd_config to check if the port settings are right
12. Lastly type sudo systemctl restart ssh to restart the SSH Service

1.4 Installing and Configuring UFW (Uncomplicated Firewall)

Install UFW

1. Type sudo apt update
2. Type sudo apt install ufw
3. Verify whether ufw was successfully installed via dpkg -l | grep ufw

Configure UFW

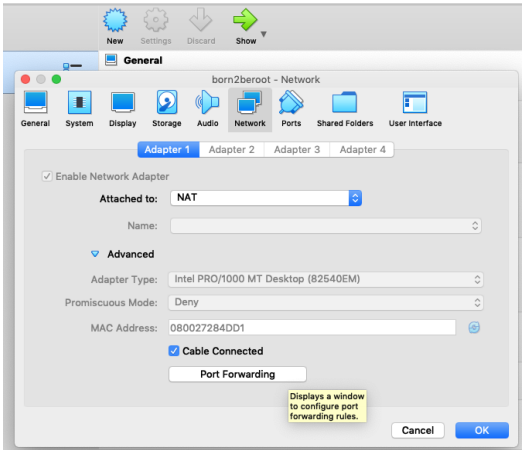
4. Type sudo ufw reset. This will clear all current rules and set UFW back to its default settings
5. Type sudo ufw allow 4242 to configure the Port Rules
6. Type sudo ufw enable to enable UFW
7. Lastly Type sudo ufw status/ sudo ufw verbose to check the status of UFW 4242 Port

*Modify the hostname sudo hostnamectl set-hostname new_hostname & reboot the system

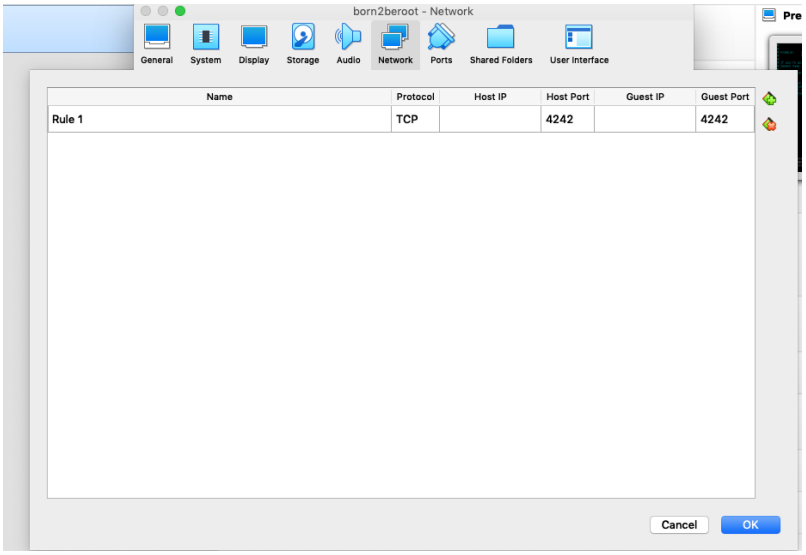
Part 2. Connecting to Server via SSH

2.1 Apply port forwarding rule on VirtualBox can be 4242:4242

1. To exit your Virtual Machine and use your mouse, press command on your Apple Keyboard and your mouse should appear
2. Go to your Virtual Box Program
3. Click on your Virtual Machine and select Settings
4. Click Network then Adapter 1 then Advanced and then click on Port Forwarding



5. Change the Host Port and Guest Port to 4242



6. Then head back to your Virtual Machine
7. Type sudo systemctl restart ssh to restart your SSH Server
8. Type sudo service sshd status to check your SSH Status

2.2 SSH into VM

9. Open an iTerm and type the following ssh <username>@<ip-address> -p 4242
   ssh lkilpela@127.0.0.1 -p 4242 # or
   ssh lkilpela@localhost -p 4242
10. In case an error occurs, then type rm ~/.ssh/known_hosts in your iTerm and then retype ssh your_username@127.0.0.1 -p 4242
11. Lastly type exit to quit your SSH iTerm Connection

* Find Ip address: ip addr show

When you run the ip addr show command, it typically displays information about all network interfaces on your system. This can include your Ethernet interface (usually named eth0, ens33, enp0s3, etc.), your loopback interface (lo), and any other network interfaces you might have.

Here's an example of what the output might look like:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen
1000
    inet 192.168.1.10/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 86378sec preferred_lft 86378sec
```

In this example, lo is the loopback interface, and eth0 is the Ethernet interface. The IP address of the Ethernet interface is 192.168.1.10.

If you're trying to SSH into the machine, you'll want to use the IP address of the Ethernet interface (or whichever interface is connected to the network you're SSHing from). If you're not sure which one to use, you might need to check your network configuration or ask your network administrator.

Part 3. User Management

3.1 Setting Password Policy

3.1.1 Password Strength

Step 1: Install the libpam-pwquality package

1. sudo apt install libpam-pwquality
   Verify whether libpam-pwquality was successfully installed
2. dpkg -l | grep libpam-pwquality

Step 2: Configure the password policy:

3. Edit the /etc/pam.d/common-password file:
   sudo vim/nano /etc/pam.d/common-password
4. Find this line. password requisite pam_deny.so or

```
password        requisite                       pam_pwquality.so retry=3
```

5. Add this to the end of that line `minlen=10 ucredit=-1 lcredit=-1 dcredit=-1 maxrepeat=3 reject_username difok=7 enforce_for_root`

   minlen=10: Sets the minimum password length to 10 characters.
   ucredit=-1: Requires at least one upper-case character in the password.
   lcredit=-1: Requires at least one lower-case character in the password.
   dcredit=-1: Requires at least one digit in the password.
   maxrepeat=3: Allows a maximum of 3 repeated characters in the password.
   reject_username: Checks if the password contains the user's name in some form.
   difok=7: Specifies that the new password should have at least 7 characters not present in the old password.
   enforce_for_root: Enforces the password policy for the root user as well.

   This line enforces a password policy requiring at least one upper-case character, one lower-case character, and one digit in all new passwords, with a minimum length of 10 characters, a maximum of 3 repeated characters, and at least 7 characters different from the old password. The policy is also enforced for the root user.

   The line should now look like this

```
password [success=1 default=ignore] pam_unix.so obscure use_authtok try_first_pass sha512
password requisite pam_pwquality.so minlen=10 ucredit=-1 lcredit=-1 dcredit=-1 maxrepeat=3 reject_username difok=7 enforce_for_root
password required pam_pwquality.so minlen=10 ucredit=-1 lcredit=-1 dcredit=-1 maxrepeat=3 reject_username enforce_for_root
```

   In this configuration:

   - The first line sets up the standard Unix password rules.
   - The second line sets up the pam_pwquality rules for non-root users, including the difok=7 rule.
   - The third line sets up the pam_pwquality rules for the root user, without the difok=7 rule.

   When a user tries to change their password, PAM will process these lines in order. If the user is not root, the second line will be processed and the third line will be skipped. If the user is root, both lines will be processed, but the difok=7 rule in the second line will be overridden by the third line.

6. Save and Exit Vim/nano

3.1.2 Password Age

7. Next type in your Virtual Machine sudo vim/nano /etc/login.defs
8. Find this part PASS_MAX_DAYS 9999 PASS_MIN_DAYS 0 PASS_WARN_AGE 7
9. Change that part to
   PASS_MAX_DAYS 30 *(To set password to expire every 30 days)*
   PASS_MIN_DAYS 2 *(To set minimum number of days between password changes to 2 days)*
   keep PASS_WARN_AGE 7 as the same *(To send user a warning message 7 days (defaults to 7 anyway) before password expiry,)*
10. Lastly type sudo reboot to reboot the change affects

*Change the password sudo passwd username

3.2 Creating a Group

1. First type sudo groupadd user42 to create a group
2. Then type sudo groupadd evaluating to create an evaluating group
3. Lastly type getent group to check if the group has been created

*Deleting a group sudo groupdel groupname

3.3 Creating a User and Assigning them Into The Group

1. First type cut -d: -f1 /etc/passwd to check all local users
2. Type sudo adduser new_username to create a username - write down your new_username, as you will need this later on.
   2.1 Type sudo usermod -aG user42 your_username or sudo adduser <username> user42
   2.2 Type sudo usermod -aG evaluating your_new_username or sudo adduser <username> evaluating
3. Type getent group user42 to check if the user is the group
4. Type getent group evaluating to check the group
5. Type groups new_username to see which groups the user account belongs to
6. Lastly type sudo chage -l new_username to check if the password rules are working in users

*Remove user from a group sudo passwd -d username groupname

Part 4. SUDO Group Configuration

4.1 Creating sudo.log

1. First type cd ~/../
2. Then type cd /var/log
3. Then type sudo mkdir -p sudo (if it already exists, then continue to the next step).
4. Then type cd sudo && touch sudo.log
5. Then type cd ~/../

Example of log entry:

```
Jan 3 15:19:06 : lkilpela : TTY=pts/0 ; PWD=/home/lkilpela ; USER=root ; COMMAND=/usr/bin/ls
```

- *Jan 3 15:19:06: This is the date and time when the command was run.*
- *lkilpela: This is the username of the person who ran the command.*

- *TTY=pts/0: This indicates the terminal where the command was run. TTY stands for teletypewriter, and pts/0 is the first pseudo-terminal slave, which is usually the terminal window you're working in.*
    - *TTY=pts/0: is basically the first terminal window you opened on your computer screen. It's like a conversation channel between you and your computer where you type commands and see the results. Ex: open a terminal window in a graphical environment (like GNOME or KDE in Linux, or Terminal.app in macOS). Pts = pseudo-terminal slave.*
    - *TTY=tty1: trefers to the first text screen you see when your computer starts up. It's like the main conversation channel between you and your computer when you're not using a graphical interface.*
- *PWD=/home/lkilpela: This shows the current directory when the command was run.*
- *USER=root: This shows that the command was run as the root user.*
- *COMMAND=/usr/bin/ls: This shows the command that was run.*
- *This log entry indicates that the user lkilpela ran the ls command as the root user in the /home/lkilpela directory at the specified date and time.*

The directory /var/log/sudo is where sudo stores its logs by default. Here's what each file or directory typically represents:

- *00: This is likely a directory for input/output logs from sudo sessions. The name of the directory is usually a timestamp and the sequence number of the command. Inside, you might find files named stdin, stdout, and stderr for input, output, and error output respectively.*
- *seq: This file is used by sudo to keep track of the sequence number for input/output logs. Each sudo session gets a unique sequence number, which is used in the naming of the session directory.*
- *sudo.log: This is the default log file where sudo logs events. This file typically contains entries for each command run with sudo, including the date and time, the username of the person who ran the command, and the command itself.*

Seq # 00000K means path 00/00/0K under folder sudo

lkilpela@lkilpela42:~$ ***sudo ls -la /var/log/sudo/00/00/0K***
total 40
drwx------ 2 root root 4096 Jan 3 16:16 .
drwx------ 22 root root 4096 Jan 3 16:16 ..
-rw------- 1 root root 75 Jan 3 16:16 log
-rw------- 1 root root 2614 Jan 3 16:16 log.json
→ log & log.json: These files contain a summary of the sudo session, including metadata like the command that was run, the user who ran it, the start and end times, etc. The log.json file is a JSON-formatted version of the log file.
-rw------- 1 root root 25 Jan 3 16:16 stderr
-rw------- 1 root root 25 Jan 3 16:16 stdin
-rw------- 1 root root 25 Jan 3 16:16 stdout
→ stderr stdin stdout: These files contain the input, output, and error output of the sudo command. They are stored in a binary format, not as plain text.
-r-------- 1 root root 1411 Jan 3 16:16 timing
→ This file contains timing information for the sudo session. It's used by the sudoreplay command to replay the session with the same timing as when it was recorded.
-rw------- 1 root root 66 Jan 3 16:16 ttyin
-rw------- 1 root root 1200 Jan 3 16:16 ttyout
→ These files contain the input and output of the terminal (tty) during the sudo session. They are also stored in a binary format.

TTY

TTY stands for "teletypewriter". In the context of modern computing, a TTY is essentially a terminal - it's a place where you type commands and see the results. When you open a command line or a terminal window on your computer, you're using a TTY.

In the context of sudo logs, ttyin and ttyout represent what was typed into the terminal (ttyin) and what the terminal displayed as a result (ttyout) during a sudo session.

4.2 Configuring Sudoers Group

1. First type sudo visudo to edit the /etc/sudoers file
2. Now edit your sudoers file to look like the following by adding in all of the defaults in the image below

```
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/bin:/sbin:/bin:/snap/bin"
Defaults        badpass_message="Incorrect password, please try again!" // Display a custom error message
Defaults        passwd_tries=3 // Limit sudo authentication attempts to 3
Defaults        logfile=”/var/log/sudo/sudo.log “// Archive each action using sudo
Defaults        log_input, log_output // Archive each action using sudo
Defaults        requiretty // Enable TTY mode
```

- Defaults env_reset: Resets the environment for the command to be run with sudo, removing any variables that could potentially affect the command's behavior.
- Defaults mail_badpass: Sends an email to the system administrator when a user enters an incorrect password for sudo.
- Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin": Sets the PATH environment variable for sudo commands. the list of places that sudo will look for commands.
  The Defaults secure_path line in the sudoers file sets the list of directories that the system will look in when you run a command with sudo.
  In simpler terms, when you type a command, your system needs to know where to find the program that handles this command. It looks in several places, specified in the PATH variable.
  When you use sudo, for security reasons, it doesn't use your usual PATH. Instead, it uses the secure_path set in the sudoers file. This line you're asking about is setting that secure_path.
- Defaults badpass_message="Incorrect password, please try again!": Sets a custom error message that's displayed when a user enters an incorrect password for sudo.
- Defaults passwd_tries=3: Limits the number of sudo authentication attempts to 3.
- Defaults logfile=”/var/log/sudo/sudo.log “: Logs all sudo actions to the specified file.
- Defaults log_input, log_output: Logs both the input to and output from sudo commands.
- Defaults requiretty: Requires that sudo be run from a real terminal (TTY). This can be a security feature, as it prevents sudo from being run from scripts or other non-interactive contexts.

Part 5. Cron/ Crontab Configuration

1. Then type apt-get install -y net-tools to install the netstat tools
2. Then type cd /usr/local/bin/
3. Then type touch monitoring.sh
4. Lastly type chmod 777 monitoring.sh

Copy Text Below onto Virtual Machine

1. Copy this text (To copy the text below, hover with your mouse to the right corner of the text below and a copy icon will appear).

```
#!/bin/bash
architecture=$(uname -a)
physical_cpu=$(lscpu | grep 'Socket(s):' | awk '{print $2}')
virtual_cpu=$(nproc)
memory_usage=$(free -m | awk '/^Mem:/{printf("%s/%sMB, (%.2f%%) \n", $3, $2,
$3/$2*100)}')
disk_usage=$(df -Bg | grep "^/dev" | grep -v "/boot" | awk '{total+=$2} {used+=$3}
END {printf "%d/%dGb (%d%%)\n", used, total, used/total*100}')
```

```
cpu_load=$(top -bn1 | grep '^%Cpu' | cut -c 9- | xargs | awk '{printf("%.1f%%"),
$1 + $2 + $3}')
last_reboot=$(who -b | awk '{print $3, $4, $5}'| xargs -I{} date -d {} +"%Y-%m-%d
%H:%M")
lvm_use=$(lsblk | grep -q 'lvm' && echo "Yes" || echo "No")
tcp_connections=$(netstat -an | grep ESTABLISHED | wc -l)
user_log=$(who | wc -l)
network=$(hostname -I | awk '{printf "%s ", $1}' ; ip addr | grep "link/ether" |
awk '{printf "(%s)", $2}')
sudo_commands_count=$(sudo grep 'COMMAND=' /var/log/sudo/sudo.log | wc -l)
wall "#Architecture: $architecture
      #CPU physical: $physical_cpu
      #vCPU: $virtual_cpu
      #Memory Usage: $memory_usage
      #Disk Usage: $disk_usage
      #CPU load: $cpu_load
      #Last boot: $last_reboot
      #LVM use: $lvm_use
      #Connections TCP: $tcp_connections ESTABLISHED
      #User log: $user_log
      #Network: IP $network
      #Sudo: $sudo_commands_count cmd"
```

1. Then open up a iTerm2 seperate from your Virtual Machine and type in iTerm ssh your_host_name42@127.0.0.1 -p 4242 and then type your password, when it asks for it.
2. Then type cd /usr/local/bin.
3. Then type nano monitoring.sh and paste the text above into the vim monitoring.sh you just created, by doing command + v on your Apple keyboard.
4. Save and Exit your monitoring.sh
   4.1 - Then type exit to exit the iTerm SSH Login.
   4.2 - Then go back to your Virtual Machine (not iTerm) and continue on with the steps below.
5. Then type sudo visudo to open your sudoers file
6. Add in this line your_username ALL=(ALL) NOPASSWD: /usr/local/bin/monitoring.sh under where its written %sudo ALL=(ALL:ALL) ALL
   - *your_username: This is the username of the user that the rule will apply to.*
   - *ALL=(ALL): The first ALL can be replaced with a hostname, meaning this rule applies no matter what machine the user is on. The =(ALL) part means the user can run commands as any user.*
   - *NOPASSWD:: This means the user is not required to enter their password to run the specified command.*
   - */usr/local/bin/monitoring.sh: This is the command that the user is allowed to run without needing to provide a password.*

   So, in simple terms, this line allows your_username to run the monitoring.sh script with sudo without having to enter a password.
7. Then exit and save your sudoers file
8. Now type sudo reboot in your Virtual Machine to reboot sudo
9. Type sudo /usr/local/bin/monitoring.sh to execute your script as su (super user)
10. Type sudo crontab -u root -e to open the crontab and add the rule
11. Lastly at the end of the crontab, type the following */10 * * * * /usr/local/bin/monitoring.sh this means that every 10 mins, this script will show

Part 6. Signature.txt (Last Part Before Defence)

⚠️ Warning: before you generate a signature number, turn off your Virtual Machine. ⚠️

1. Open iTerm and type cd
2. Then type cd sgoinfre/students/<your_intra_username>/VirtualBox VMs
3. Type shasum debian.vdi or whatever your Virtual Machine is called (This can take from a few seconds to 5 mins).
4. Copy the output number and create a signature.txt file and paste that number in the file.
5. Now you submit the signature.txt file with the output number in it.

6f273e3a65aa39be408cb3e226fe9e5e85e537d2 debian.vdi

**debian.vdi* is a VirtualBox Disk Image file. It's a file format used by VirtualBox software, a free and open-source virtualization software, to store the hard disk of a virtual machine (VM).

In this case, `debian.vdi` likely represents a VM that has Debian (a popular Linux distribution) installed. The `.vdi` file contains the operating system, software, and files that are installed on the virtual machine.

COMMAND & DEFINITION

1. Check UFW status	sudo ufw status
2. Check SSH status	sudo systemctl status ssh
3. Create user	sudo adduser <new_username>
4. Delete user	sudo cut -d: -f1 /etc/passwd
5. Check user list	getend passwd | cut -d: -f1
6. Create group	sudo groupadd <groupname>
7. Assign group	sudo usermod -aG <groupname> <username>
8. Check group list	getend group | cut -d: -f1
9. Check passwd age	sudo chage -l username
10. Check user belongs to group	getent group evaluating
11. Check hostname	hostnamectl
12. Modify hostname	hostnamectl set-hostname <new_hostname>
13. Change hostname in file	sudo nano /etc/hosts
14. Check the partitions	lsblk
15. Check sudo installed	dpkg -l | grep sudo –
16. Assign the evaluator's username to the "sudo" group	sudo usermod -aG <groupname> <username>
17. Verify that the " /var/log/sudo/" folder exists and has at least one file.	cd /var/log/sudo/
List files in folder	ls
Display sudo.log	cat sudo.log
18. Check UFW installed	dpkg -l | grep ufw –
19. List the active rule in UFW	sudo ufw status numbered
20. Add a new rule to open port 8080	sudo ufw allow 8080
21. Check that it has been added to the active rules	sudo ufw status numbered
22. Delete this new added rule	sudo ufw delete <rule_number>
23. Check SSH installed	dpkg -l | grep ssh –
24. Open the crontab and add the rule	sudo crontab -u root -e

***dpkg -l lists all the packages**

dpkg is the package management command in Debian-based systems. It's used to install, remove, and manage software packages.
The -l option stands for "list". So dpkg -l lists all the packages.
When you run dpkg -l, you'll see a table with columns. Here's what they mean:

- The first column shows the desired action: i for install, r for remove, p for purge.
- The second column shows the current status: i for installed, c for config-files, h for half-installed, u for unpacked, f for half-configured, w for triggers-awaited, t for triggers-pending.
- The third column shows errors, if any.
- The fourth column shows the package name.
- The fifth column shows the version of the package.
- The sixth column shows the architecture of the package (like amd64 for 64-bit systems).
- The last column shows a brief description of the package.

***Ssh vs sshd**

ssh and sshd are both related to the SSH (Secure Shell) protocol, which is used for secure remote login and other secure network services over an insecure network.

- ssh: This is the client command used to start a new SSH session. For example, you might use ssh username@hostname to start a new SSH session with a remote server.
- sshd: This stands for SSH daemon, which is the server side of the SSH connection. The sshd service listens for incoming SSH requests and handles the authentication and establishment of SSH sessions.

In short, when you use ssh to connect to a remote server, you're connecting to the sshd service running on that server.

***grep = print lines that match patterns, searches for PATTERNS in each FILE**

grep is a command-line utility for searching plain-text data sets for lines that match a regular expression. Its name comes from the ed command g/re/p (globally search a regular expression and print).

```
grep 'pattern' filename
```

Its name comes from the ed command g/re/p (globally search a regular expression and print). This command will search for the string 'pattern' within the file named 'filename' and print out any lines that contain that pattern.

Here are a few commonly used options with grep:

-i: Ignore case (so, for example, 'a' will match 'A').
-r or -R: Recursively search directories.
-v: Invert match, i.e., only show lines that do not match the pattern.
-l: Only print filenames of files that contain the match.
-n: Show line numbers along with lines that match the pattern.
Remember, grep is case sensitive. It treats lowercase and uppercase as different characters, unless the -i option is used.

***wc -l = print the newline counts**

wc is a command-line utility in Unix/Linux operating systems that displays the number of lines, words, and bytes contained in files. The name wc stands for "word count".

```
wc filename
```

This command will output three numbers and the filename. The first number is the number of lines in the file, the second is the number of words, and the third is the number of bytes.

Here are some commonly used options with wc:

-l: Print the number of lines in a file.
-w: Print the number of words in a file.
-c or -m: Print the number of bytes in a file. -c counts bytes, while -m counts characters.
-L: Print the length of the longest line in a file.
For example, wc -l filename will only output the number of lines in the file.

***awk = pattern scanning and text processing language**

awk is a powerful command-line tool in Unix/Linux for processing text files, particularly useful for data extraction and reporting. It's a complete programming language optimized for text processing.

```
awk '/pattern/ {print $0}' filename
```

This command will search for the string 'pattern' within the file named 'filename' and print out any lines ($0 refers to the entire line) that contain that pattern.

Here are some commonly used features with awk:

$1, $2, ..., $n: These are used to refer to the respective fields in a line. For example, $1 refers to the first field.
FS: This built-in variable is used to set the field separator. The default is white space.
NR: This built-in variable keeps a current count of the number of input lines.
NF: This built-in variable contains the number of fields within the current record.
BEGIN and END: These special patterns allow you to specify actions to take before the first input line is read and after the last input line is read, respectively.
For example, awk -F: '{print $1}' /etc/passwd will print the usernames (the first field) from the /etc/passwd file, where fields are separated by a colon (:).

***shasum = Print or check SHA Checksums**

SHA = SHA, or Secure Hash Algorithm, is like a digital fingerprint for data. It takes input (like a file or a message) and returns a fixed-size string of bytes, which is the "fingerprint". Even a small change in the input will produce a completely different fingerprint. This is commonly used to check if data has been altered or corrupted.

***crontab** is a program in Unix-like operating systems that users use to schedule jobs to run at fixed times, dates, or intervals.

To edit or create a new crontab file, you can use the command crontab -e. This will open the crontab file in your default text editor.

A crontab file has the following format:

```
*     *     *   *    *        command to be executed
-     -     -   -    -
|     |     |   |    |
|     |     |   |    +----- day of the week (0 - 6) (Sunday=0)
|     |     |   +------- month (1 - 12)
|     |     +--------- day of the month (1 - 31)
|     +----------- hour (0 - 23)
+------------- min (0 - 59)
```

For example, if you want to run a script at 5 a.m every day, you would add the following line to your crontab file:

```
0 5 * * * /path/to/your/script.sh
```

If you want to redirect the output of the script to a file, you can modify the crontab entry like this:

```
0 5 * * * /path/to/your/script.sh > /path/to/log.txt
```

This will redirect the standard output of the script to log.txt. If you also want to redirect the standard error, you can do so like this:

```
0 5 * * * /path/to/your/script.sh > /path/to/log.txt 2>&1
```

This will redirect both the standard output and the standard error of the script to log.txt.

*Appamor module is loaded

```
sudo apparmor_status
```

*ss -tunlp command is used to display detailed information about open network sockets on a Linux system. Specifically:

-t: Show TCP sockets.
-u: Show UDP sockets.
-n: Show numerical addresses (don't resolve hostnames).
-l: Show only listening sockets.
-p: Show the process that owns each socket.
So, when you run ss -tunlp, it provides a list of all listening TCP and UDP sockets along with the associated process and its process ID.

Here's a breakdown of the output columns:

- Local Address (LADDR:LPORT): The local address and port where the socket is bound.
- Peer Address (RADDR:RPORT): The remote address and port if connected to a remote system.
- State: The state of the socket (e.g., LISTEN, ESTAB for established connections).
- PID/Program name: The Process ID (PID) and the name of the program or service that owns the socket.

CONCEPT

1. Linux:

Here are the key points you need to understand about the basics of Linux:

a. Command Line: The command line, also known as the terminal, is a text-based interface to the system. You enter commands as text and receive text-based feedback. It's a powerful tool that can perform complex tasks quickly and automate repetitive tasks.

b. File System: Linux uses a hierarchical file system, with the root (/) directory at the top. Key directories include /home (user directories), /etc (configuration files), /var (variable data like logs), /bin and /usr/bin (system and user binaries), and /lib and /usr/lib (libraries).

c. Basic Commands:

- ls: Lists the contents of a directory.
- cd: Changes the current directory.
- mv: Moves or renames files and directories.
- cp: Copies files and directories.
- rm: Removes files and directories.
- pwd: Prints the current working directory.
- cat: Concatenates and displays file content.
- echo: Outputs the strings it is being passed as arguments.
- man: Displays the user manual of any command that it is given.

d. Installing and Configuring a Linux Distribution: This process varies depending on the distribution. You'll typically need to download an ISO file, write it to a bootable medium (like a USB drive), boot from that medium, and then follow the installation prompts. Configuration can involve tasks like setting up users, installing software, and configuring services.

To learn these topics, you can use online tutorials, books, and other resources. Practice is also important: try using the command line for tasks you'd normally do with a graphical interface, and try setting up a Linux system in a virtual machine to get a feel for the installation and configuration process.

2. Virtual Machine (VM):

Learn how to create a VM using software like VirtualBox or UTM. Understand how to configure the VM's settings, like allocating RAM and hard drive space.

➢ **UTM**

UTM is a virtual machine host for iOS. It allows you to run Windows, Android, and Linux on your iOS device without needing to jailbreak it. Here's a basic guide on how to create a virtual machine (VM) using UTM:

- Download UTM: First, you'll need to download the UTM app from the App Store on your iOS device.
- Create a New VM: Open UTM and tap the "+" button to create a new VM.
- Configure the VM: You'll need to provide some basic information about the VM:

    - Name: This can be anything you like.
    - Architecture: This depends on the operating system you want to install. For example, for Windows, you might choose x86_64.
    - System: Again, this depends on the operating system. For Windows, you might choose pc.
- Configure the Drives: You'll need to add a drive for the operating system:
    - Tap "Drives" > "+" > "Path", then select your ISO file.
    - Set the "Interface" to ide.
- Configure the Network: If you want network access, you'll need to configure the network settings:
    - Set "Network Card" to e1000.
    - Set "Network Mode" to Emulated.
- Start the VM: Once you've configured everything, you can start the VM by tapping "Save" and then tapping on the VM in the list.

Remember, the exact steps and options will depend on your specific situation and the operating system you're installing. Always refer to the UTM documentation for the most accurate information.

Please note that while UTM is a powerful tool, running a VM on an iOS device can be resource-intensive and may not provide the same level of performance as running the operating system on a dedicated machine.

➢ **VirtualBox**

Here's a step-by-step guide on how to create a Virtual Machine (VM) using VirtualBox:

1. **Download and Install VirtualBox:** You can download VirtualBox from the official website. Once downloaded, run the installer and follow the prompts to install VirtualBox on your system.

2. **Create a New Virtual Machine:**
   - Open VirtualBox and click on "New" to create a new VM.
   - Give your VM a name, and choose the type and version of the operating system you'll be installing.
   - Click "Next" to proceed.

3. **Allocate RAM:**
   - You'll be asked to allocate memory (RAM) to your VM. The amount you choose will depend on the requirements of the operating system you're installing and the amount of RAM available on your host machine.
   - After allocating RAM, click "Next".

4. **Create a Virtual Hard Disk:**
   - You'll be asked to create a virtual hard disk for your VM. Choose "Create a virtual hard disk now" and click "Create".
   - Choose the hard disk file type. VDI (VirtualBox Disk Image) is the default and will work for most users.
   - Choose whether you want the virtual hard disk to be dynamically allocated (it will only use space on your physical hard drive as it fills up, up to the maximum size you set) or fixed size (it will immediately take up the full amount of space you set).
   - Set the size of the virtual hard disk. The size you choose will depend on the requirements of the operating system you're installing and the amount of free space on your host machine.

5. **Configure VM Settings:**
   - Once the VM is created, you can further configure its settings by selecting the VM and clicking "Settings". Here you can adjust settings like the number of CPU cores the VM can use, the video memory, the boot order, and more.

6. **Install the Operating System:**
   - With the VM created and configured, you can now install the operating system. You'll need an installation media (like an ISO file) for the operating system you want to install.
   - Start the VM and follow the prompts to install the operating system.

Remember, the exact steps may vary depending on the version of VirtualBox you're using and the operating system you're installing. Always refer to the documentation for your specific situation.

3. Partitioning:

A partition is a division of the storage space of a hard drive. Each partition functions as if it were a separate hard drive. Partitions are used for several reasons:

1. **Organization:** Partitions can help keep your data organized. For example, you can keep your operating system files separate from your personal files by putting them on different partitions.

2. **Multiple Operating Systems:** If you want to install more than one operating system on a single computer (a configuration known as dual-booting or multi-booting), you'll need a separate partition for each OS.

3. **Security and Stability:** Keeping system files separate from user files can help prevent user data from filling up the drive and crashing the system. It can also make it easier to recover data if one partition fails.

During the Linux installation process, you'll have the option to partition your hard drive. Here's a general process:

1. Boot from your Linux installation media (like a USB stick or DVD).
2. Follow the prompts until you reach the partitioning step. The exact details will depend on your Linux distribution.
3. You'll usually see options to erase the disk and install Linux (which creates the necessary partitions automatically), or to manually create partitions. If you choose to create partitions manually, you'll need to at least create a root partition (`/`), and typically a swap partition as well. You may also want to create separate partitions for `/home`, `/boot`, and other directories, depending on your needs.
4. After creating the partitions, continue with the installation process.

Remember, partitioning a drive will erase all data on it, so make sure to back up any important data before you start. Also, the exact steps for partitioning will depend on the Linux distribution you're installing and the tool you're using. Always refer to the documentation for your specific situation.

4. LVM (Logical Volume Manager):

Learn what LVM is and why it's used. Understand how to create and manage logical volumes.

LVM, or Logical Volume Manager, is a storage device management technology that gives you more flexibility in managing disk space. Instead of being restricted to physical disk partitions, LVM allows you to create logical volumes that can span multiple physical disks and partitions.

Here's why LVM is used:

1. **Flexibility:** You can resize logical volumes as needed, even while they're in use. This is useful if your space requirements change over time.
2. **Snapshots:** LVM can create snapshots, which are point-in-time copies of logical volumes. This can be useful for backups or for testing changes before applying them to the live system.
3. **Spanning:** A single logical volume can span multiple physical disks, allowing you to use all your storage as one big disk.

Here's a basic guide on how to create and manage logical volumes:

1. **Create a Physical Volume:** First, you need to create one or more physical volumes. These are the actual disks or partitions that will hold your data. Use the `pvcreate` command to create a physical volume:

```
pvcreate /dev/sdb1
```

Replace `/dev/sdb1` with the path to your disk or partition.

2. **Create a Volume Group:** Next, create a volume group. This is a pool of disk space that you can allocate to logical volumes. Use the `vgcreate` command to create a volume group:

```
pvcreate /dev/sdb1
```

Replace `myvg` with the name of your volume group, and `/dev/sdb1` with your physical volume.

3. **Create a Logical Volume:** Now you can create a logical volume from the space in your volume group. Use the `lvcreate` command to create a logical volume:

```
lvcreate -L 10G -n mylv myvg
```

This creates a logical volume named `mylv` in the volume group `myvg`, with a size of 10 GB.

4. **Format and Mount the Logical Volume:** Before you can use your logical volume, you need to format it with a file system and mount it:

```
mkfs.ext4 /dev/myvg/mylv
mount /dev/myvg/mylv /mnt/mylv
```

This formats the logical volume with the ext4 file system and mounts it at `/mnt/mylv`.

Remember, the exact commands and options will depend on your specific situation. Always refer to the LVM documentation or man pages (`man lvm`) for the most accurate information.

5. Sudo:

Understand what sudo is and why it's used. Learn how to configure sudo permissions for different users.

`sudo` stands for "superuser do". It's a command used in Unix and Linux systems that allows you to run programs with the security privileges of another user (by default, the superuser, or root).

Here's why `sudo` is used:

1. **Security:** Running as the superuser all the time can be dangerous, as any mistake can potentially cause system-wide damage. With `sudo`, you can work as a regular user and only elevate privileges when necessary.

2. **Auditability:** `sudo` logs all commands and attempts to use it, providing a trace of who did what.

3. **Control:** `sudo` allows fine-grained control over who can do what. You can configure `sudo` to allow certain users to run specific commands as root or another user.

To configure `sudo` permissions, you edit the `sudoers` file. This is typically done with the `visudo` command, which opens the `sudoers` file in a safe way (it checks for syntax errors before saving).

Here's a basic guide on how to give a user `sudo` permissions:

1. Open the `sudoers` file by running `sudo visudo` in the terminal.

2. In the `sudoers` file, you'll see a line like this:

```
root    ALL=(ALL:ALL) ALL
```

This means that the root user can run any command on any host as any user.

3. To give `sudo` permissions to a user, add a line similar to the root line, replacing `root` with the username. For example, to give `sudo` permissions to a user named `username`, you would add:

```
username    ALL=(ALL:ALL) ALL
```

4. Save and close the file. In `vi` or `vim`, you can do this by typing `:wq` and pressing Enter.

Now, `username` can run commands with `sudo`. They'll be asked for their password the first time they run a command with `sudo` in each session.

Remember, giving a user `sudo` permissions gives them a lot of power over the system, so be careful who you give `sudo` access to. Always refer to the `sudo` and `visudo` man pages (`man sudo`, `man visudo`) for more information.

6. User and Group Management:

Learn how to add, modify, and delete users and groups in Linux. Understand the difference between regular users and superusers.
In Linux, you can manage users and groups using command-line tools. Here's a basic guide:

- ➢ Adding Users:

Use the useradd or adduser command to add a new user. For example:

```
sudo useradd username
```

This creates a new user named username. You can set a password for the new user with the passwd command:

```
sudo passwd username
```

➢ Modifying Users:

Use the usermod command to modify an existing user. For example, to add a user to a group:

```
sudo usermod -aG groupname username
```

This adds username to groupname.

➢ Deleting Users:

Use the userdel command to delete a user:

```
sudo userdel username
```

This deletes username. To remove the user's home directory as well, use the -r option:

➢ Adding Groups:

Use the groupadd command to add a new group:

```
sudo groupadd groupname
```

➢ Deleting Groups:

Use the groupdel command to delete a group:

```
sudo groupdel groupname
```

➢ Regular Users vs Superusers:

A regular user is a non-privileged user that can use the system but doesn't have the power to perform administrative tasks. A superuser (or root user) has unrestricted access to the system and can perform any operation.

The sudo command allows regular users to perform actions with superuser privileges, as configured in the sudoers file. This allows for better security and auditability, as users can work as regular users most of the time and only elevate their privileges when necessary.
Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the man pages (man useradd, man usermod, etc.) for the most accurate information.

7. SSH (Secure Shell):

Understand what SSH is and why it's used. Learn how to configure an SSH server and how to connect to a remote server using SSH.

SSH, or Secure Shell, is a protocol used to securely connect to a remote server. It's commonly used for managing servers, executing commands remotely, and for secure file transfers with SCP or SFTP.

Here's why SSH is used:

1. **Security:** All data sent over an SSH connection, including passwords, is encrypted, making it difficult for anyone to intercept your data.
2. **Authentication:** SSH uses public key cryptography for authentication, which is more secure than password-based authentication.
3. **Functionality:** SSH allows you to run commands on a remote server, which is essential for server administration. It also supports tunneling, port forwarding, and file transfers.

Here's a basic guide on how to configure an SSH server and connect to it:

Configuring an SSH Server:

1. Install the OpenSSH server. On Ubuntu, you can do this with the following command:

```
sudo apt-get install openssh-server
```

2. The SSH server should start automatically after installation. You can check its status with:

```
sudo systemctl status ssh
```

3. The server's configuration file is located at `/etc/ssh/sshd_config`. You can edit this file to change the server's settings. For example, to change the port that the server listens on, you can modify the `Port` line in the file.

Connecting to an SSH Server:

1. From a client machine, you can connect to the SSH server with the `ssh` command:

```
ssh username@hostname
```

Replace `username` with your username on the server, and `hostname` with the server's hostname or IP address.

2. The first time you connect to a server, you'll be asked to verify the server's identity. If you accept, the server's public key will be saved and you won't be asked again.

3. After connecting, you can run commands on the server as if you were sitting at the server's console.

Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the OpenSSH documentation or man pages (`man ssh`, `man sshd`) for the most accurate information.

8. Firewall:

Learn what a firewall is and why it's used. Understand how to configure a firewall using tools like UFW or iptables.

A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. It establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.

Here's why a firewall is used:

1. **Security:** A firewall can prevent unauthorized access to or from a private network, enhancing the security of devices connected to this network.
2. **Control:** It allows you to control the flow of traffic to and from your network based on rules that you define.
3. **Logging:** Firewalls can log network activity, helping you understand traffic patterns and detect suspicious behavior.

Here's a basic guide on how to configure a firewall using UFW (Uncomplicated Firewall), a user-friendly front-end for iptables:

Installing UFW:

On Ubuntu, you can install UFW with the following command:

```
sudo apt-get install ufw
```

Enabling and Disabling UFW:

To enable UFW, use the `ufw enable` command:

```
sudo ufw enable
```

To disable UFW, use the `ufw disable` command:

```
sudo ufw disable
```

Setting UFW Rules:

To allow traffic on a specific port, use the `ufw allow` command. For example, to allow traffic on port 22 (the default SSH port):

```
sudo ufw allow 22
```

To deny traffic on a specific port, use the `ufw deny` command. For example, to deny traffic on port 25 (the default SMTP port):

```
sudo ufw deny 25
```

Checking UFW Status and Rules:

To check the status of UFW and see the current rules, use the `ufw status` command:

```
sudo ufw status
```

Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the UFW or iptables documentation or man pages (`man ufw`, `man iptables`) for the most accurate information.

9. Monitoring:

Learn how to use tools like top, htop, netstat, etc. to monitor system resources. Understand what information these tools provide and how to interpret it.
These tools are used in Linux to monitor system resources:

1. **top:** This is a real-time system monitor that displays a dynamic view of the processes running in a system. It provides a quick overview of system performance, showing information like CPU usage, memory usage, load average, and more.

To use `top`, simply type `top` in your terminal:

```
top
```

The key metrics to look at are:

- **PID:** Process ID
- **USER:** User running the process
- **PR:** Process priority
- **NI:** The nice value of the process
- **VIRT:** Virtual memory used by the process
- **RES:** Resident memory used by the process
- **SHR:** Shared memory of the process
- **%CPU:** CPU usage of the process
- **%MEM:** Memory usage of the process
- **TIME+:** Total CPU time the process has taken
- **COMMAND:** The command that initiated the process

2. **htop:** This is an interactive process viewer, similar to `top`, but it provides a more user-friendly interface and allows for easier navigation.

To use `htop`, you might need to install it first:

```
sudo apt-get install htop
```

Then, you can run it by typing `htop` in your terminal:

```
htop
```

`htop` shows similar information to `top`, but it also includes color coding for quick visual analysis and it allows you to scroll vertically and horizontally.

3. **netstat:** This tool displays network connections, routing tables, interface statistics, and more. It's useful for monitoring network activity.

To use `netstat`, type `netstat` in your terminal:

```
netstat
```

The key metrics to look at are:

- **Proto:** The protocol (tcp, udp)
- **Recv-Q:** The count of bytes not copied by the user program connected to this socket
- **Send-Q:** The count of bytes not acknowledged by the remote host
- **Local Address:** The local address and port number
- **Foreign Address:** The foreign address and port number

- **State:** The current state of the connection

Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the man pages (`man top`, `man htop`, `man netstat`) for the most accurate information.

10. Scripting:

Shell scripting is a powerful tool that allows you to automate tasks in Linux. A shell script is a text file that contains a sequence of commands for a Unix-based operating system.

Here's a basic guide on how to write a shell script:

1. **Create a new script file:** Use a text editor to create a new file. The file should have the `.sh` extension. For example:

```
nano script.sh
```

2. **Start with a shebang:** The first line of your script should be a shebang (`#!`) followed by the path to the shell that you want to interpret the script. For a bash script, this would be:

```
#!/bin/bash
```

3. **Write commands:** After the shebang, you can start writing commands as if you were typing them into the terminal. For example, to print "Hello, world!" you would use the `echo` command:

```
echo "Hello, world!"
```

4. **Make the script executable:** After saving the script, you need to make it executable with the `chmod` command:

```
chmod +x script.sh
```

5. **Run the script:** Now you can run the script like this:

```
./script.sh
```

Here's an example of a script that uses commands to get system information:

```
#!/bin/bash

# Print the hostname
echo "Hostname: $(hostname)"

# Print the system uptime
echo "Uptime: $(uptime)"

# Print the current date and time
echo "Current date and time: $(date)"

# Print the disk usage
echo "Disk usage:"
df -h

# Print the memory usage
echo "Memory usage:"
```

```
free -h
```

This script uses the `hostname`, `uptime`, `date`, `df`, and `free` commands to print various system information. The `$(command)` syntax is used to insert the output of a command into a string.

Remember, shell scripting can be much more complex than this, with features like variables, conditionals, loops, and functions. Always refer to the bash or shell scripting documentation or man pages (`man bash`) for the most accurate information.

11. SELinux:

Understand what SELinux is and why it's used. Learn how to enable, disable, and configure SELinux.

SELinux, or Security-Enhanced Linux, is a security module for the Linux kernel that provides a mechanism for supporting access control security policies.

Here's why SELinux is used:

1. **Mandatory Access Controls:** SELinux implements Mandatory Access Controls (MAC) to confine programs to a limited set of resources and operations, based on defined policies.
2. **Defense in Depth:** Even if an attacker manages to exploit a vulnerable program, SELinux can help prevent them from accessing other resources or executing certain operations.
3. **Policy Enforcement:** SELinux enforces policies that you define, giving you more control over your system's security.

Here's a basic guide on how to enable, disable, and configure SELinux:

Check SELinux Status:

You can check the current status of SELinux with the `sestatus` command:

```
sestatus
```

Enable SELinux:
To enable SELinux, you need to edit the `/etc/selinux/config` file and set `SELINUX=enforcing`. Then, reboot your system for the changes to take effect.

```
sudo nano /etc/selinux/config
```

In the file, change the line to `SELINUX=enforcing`.

Disable SELinux:

To disable SELinux, edit the `/etc/selinux/config` file and set `SELINUX=disabled`. Then, reboot your system for the changes to take effect.

In the file, change the line to `SELINUX=disabled`.

Configure SELinux:

Configuring SELinux involves defining policies that control how programs interact with resources. This is a complex task that requires a deep understanding of your system and the principles of access control.

One tool that can help with this is `audit2allow`, which generates SELinux policy modules based on logs of denied operations.

Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the SELinux documentation or man pages (`man selinux`, `man audit2allow`) for the most accurate information..

12. Cron:

Learn what cron is and why it's used. Understand how to schedule tasks using cron and how to write cron expressions.

Cron is a time-based job scheduler in Unix-like operating systems. Users can schedule jobs (commands or scripts) to run at specific times or on specific days.

Here's why cron is used:

1. Automation: Cron allows you to automate repetitive tasks.
2. Scheduling: You can schedule tasks to run during off-peak hours to improve system performance.
3. Maintenance: System maintenance tasks like backups, cleaning up disk space, or updating software can be scheduled with cron.

Here's a basic guide on how to schedule tasks using cron:

Edit the Crontab:

The `crontab` (short for "cron table") is a file that contains a list of jobs that cron will execute. You can edit your user's crontab with the `crontab -e` command:

```
crontab -e
```

Write a Cron Expression:

A cron expression is a string of fields separated by spaces. Each field represents a unit of time. The fields are:

1. Minute (0 - 59)
2. Hour (0 - 23)
3. Day of the month (1 - 31)
4. Month (1 - 12)
5. Day of the week (0 - 7) (Sunday = 0 or 7)

Here's an example of a cron expression:

```
30 2 * * * /path/to/command
```

This will run `/path/to/command` at 2:30 AM every day.

Special Characters:

- Asterisk (`*`): This represents "all" or "every". For example, an asterisk in the "day of the week" field means "every day of the week".

- Comma (`,`): This is used to specify a list. For example, `1,15` in the "day of the month" field means "the 1st and 15th of the month".
- Hyphen (`-`): This is used to specify a range. For example, `2-4` in the "day of the week" field means "Tuesday to Thursday".
- Slash (`/`): This is used to specify a step value. For example, `*/10` in the "minute" field means "every 10 minutes".

Remember, the exact commands and options will depend on your specific situation and the Linux distribution you're using. Always refer to the cron and crontab documentation or man pages (`man cron`, `man crontab`) for the most accurate information.

DEFINITION

-partition

a partition is a division of a hard disk drive (HDD) or solid-state drive (SSD). Each partition functions as if it were a separate disk drive. Partitions are used to manage the storage space of a drive more efficiently, and they can have different file systems.

When you install an operating system, you can choose to use the entire disk for the system, or you can divide the disk into multiple partitions. For example, you might have one partition for the operating system and another for your personal files. This can make it easier to manage your files and can also provide some benefits in terms of system performance and data recovery.

In a Linux system, you'll often see at least two partitions: one for the root filesystem (/) and one for swap. The root filesystem contains all the files used by the operating system, while the swap partition is used as extra memory by the system. You might also see separate partitions for other directories, such as /home or /boot.

Here's a simple example of what a partitioned disk might look like:

```
|-----------------|-----------------|-----------------|
|    / (root)     |      /home      |      swap       |
|-----------------|-----------------|-----------------|
```

Each partition is like a separate disk drive, with its own filesystem and its own set of files.

-hard disk drive (HDD) & solid-state drive (SDD)

A Hard Disk Drive (HDD) and a Solid-State Drive (SSD) are two types of storage devices used in computers.

Hard Disk Drive (HDD): This is the traditional spinning hard drive that consists of one or more platters to which data is written using a magnetic head, all inside of an air-sealed casing. Data is stored on the hard drive in a non-volatile way, meaning it remains stored even when the computer is turned off. HDDs are known for their large storage capacity and lower cost, but they're also slower and more prone to mechanical failures compared to SSDs.

Solid-State Drive (SSD): This is a newer type of storage device that uses flash-based memory, which is significantly faster than a hard disk drive. Unlike HDDs, SSDs have no moving parts, which gives them advantages over HDDs such as faster read and write times, quieter operation, and greater durability. However, SSDs are more expensive than HDDs for the same amount of storage.

In many modern computer systems, it's common to see a combination of both - an SSD for the operating system and frequently used files (to take advantage of the speed), and an HDD for bulk storage of less frequently used files.

Ổ cứng SSD (Solid State Drive) và HDD (Hard Disk Drive) là hai loại ổ cứng phổ biến được sử dụng trong máy tính và các thiết bị lưu trữ khác. Dưới đây là một số khác biệt chính giữa chúng:

Cấu trúc vật liệu:

SSD: Sử dụng bộ nhớ flash không di động (chẳng hạn như NAND) để lưu trữ dữ liệu.
HDD: Sử dụng các đĩa quay và đầu đọc/ghi để lưu trữ dữ liệu trên các bề mặt từ tính.
Tốc độ truy cập và đọc/ghi:

SSD: Nhanh hơn đáng kể so với HDD. Thời gian truy cập và đọc/ghi là rất ngắn, do không có các bộ cơ cấu cơ học.
HDD: Chậm hơn so với SSD vì cần thời gian để đưa đầu đọc/ghi đến vị trí đúng trên đĩa quay.
Trọng lượng và kích thước:

SSD: Nhẹ hơn và thường nhỏ gọn hơn, không chứa các bộ phận cơ học lớn.
HDD: Thường nặng hơn và có kích thước lớn hơn do có các đĩa quay và cơ cấu cơ học.
Tiêu thụ năng lượng:

SSD: Tiêu thụ năng lượng thấp hơn do không có các bộ cơ cấu cơ học chạy liên tục.
HDD: Tiêu thụ năng lượng cao hơn do cần quay đĩa và đưa đầu đọc/ghi.
Độ tin cậy và bền bỉ:

SSD: Thường ít bị hỏng hơn trong điều kiện rung động hoặc va đập do không có bộ phận cơ học chạy.
HDD: Có thể bị hỏng dễ dàng hơn do có các bộ cơ cấu cơ học.
Giá cả:

SSD: Thường đắt hơn so với HDD với dung lượng tương đương.
HDD: Thường rẻ hơn và có sẵn trong dung lượng lớn hơn với giá trung bình mỗi đơn vị dung lượng thấp hơn.
Tùy thuộc vào nhu cầu sử dụng và ngân sách, người dùng có thể lựa chọn giữa SSD và HDD để đáp ứng yêu cầu lưu trữ và hiệu suất của họ.

-apt & aptitude

apt and aptitude are both command-line package management interfaces in Debian-based systems like Ubuntu. They are used to manage packages (software), allowing you to install, remove, update, and manage the software on your system. Here are some differences between the two:

User Interface: aptitude has a text-based user interface if run without any commands, which apt doesn't have. This can make it easier to browse packages, manage dependencies, and perform other tasks.

Handling of Dependencies: aptitude is more aggressive in resolving dependencies. It tries to handle dependencies more intelligently and offers more options when it comes to resolving dependencies.

Logging: aptitude logs actions (like installations and removals) in /var/log/aptitude, while apt doesn't have a dedicated log file. However, apt actions can be found in /var/log/dpkg.log.

Package Suggests: aptitude installs recommended packages as default which apt doesn't.

Search: aptitude provides more detailed search functionality compared to apt.

In general, both apt and aptitude are capable tools, and the choice between them often comes down to personal preference. For most users, apt's simpler interface and command structure is more than sufficient.

-Debian

Debian is a free and open-source operating system that uses the Linux kernel, and also the BSD kernel for its Debian GNU/kFreeBSD variant. It's one of the oldest and most respected Linux distributions, known for its robustness and rich repository of software packages.

Here are some key features of Debian:

- Stability and Reliability: Debian is known for its stability and reliability, which makes it a popular choice for servers.
- Software Repository: Debian has one of the largest software repositories with over 51,000 packages.
- Release Cycle: Debian does not follow a fixed release schedule, but releases a new version when it's considered ready. This contributes to its reputation for stability.
- Community-Driven: Debian is a community-driven project, with development and maintenance done by volunteers around the world.
- The Debian Social Contract: This is a document that outlines the project's commitment to free software and its users.
- Used as a Base for Other Distributions: Many popular Linux distributions, including Ubuntu and its derivatives, are based on Debian.

Debian can be used as a desktop as well as a server operating system. It supports a wide range of computer architectures, from x86 and x86-64 to ARM, MIPS, and more.

-SELinux & AppArmor:

SELinux (Security-Enhanced Linux) and AppArmor (Application Armor) are both Linux security modules that provide mandatory access control (MAC). They help limit the potential damage from vulnerabilities in applications and system services. Here's a brief comparison:

SELinux:

- Developed by the United States National Security Agency (NSA) and integrated into the Linux kernel.
- Uses labels and policy rules to determine access controls.
- Provides fine-grained controls and flexibility.
- Can be complex to set up and manage due to its complexity and extensive capabilities.
- Used by default in distributions like Fedora, CentOS, and Red Hat.

AppArmor:

- Developed by Novell and integrated into the Linux kernel.
- Uses file paths and network access controls to determine access controls.
- Easier to set up and manage, with a simpler policy language.
- Less flexible and fine-grained compared to SELinux.
- Used by default in distributions like Ubuntu and openSUSE.

- The choice between SELinux and AppArmor often depends on your specific needs, the distribution you're using, and your comfort level with managing each tool. Both provide significant security benefits over running without a MAC system.

-linux kernel, bsd kernel, gnu/kfreebsd variant

Linux Kernel: The Linux kernel is the core of the Linux operating system. It's responsible for managing the system's resources, and providing a way for applications to interact with the hardware. The Linux kernel was created by Linus Torvalds in 1991, and it's used in a wide variety of operating systems, from desktops and servers (like Debian and Ubuntu) to embedded systems and supercomputers.

BSD Kernel: The BSD kernel is the core of the BSD (Berkeley Software Distribution) operating system. It's a derivative of the Unix operating system that was developed at the University of California, Berkeley. There are several versions of BSD, including FreeBSD, OpenBSD, and NetBSD, each with its own kernel. The BSD kernel is known for its robustness, performance, and security features.

GNU/kFreeBSD Variant: Debian GNU/kFreeBSD is a variant of the Debian operating system that uses the FreeBSD kernel instead of the Linux kernel, but continues to use most of the GNU operating system components. This provides a unique blend of FreeBSD's advanced networking, storage, and security features with the familiar Debian environment. It's an official Debian distribution, alongside Debian GNU/Linux.

-Virtual Machine

A Virtual Machine (VM) is a software emulation of a computer system. It runs on a physical computer known as the host machine and provides the functionality of a physical computer, known as the guest machine.

Here are some key points about virtual machines:

1. **Isolation:** Each VM is isolated from the host machine and other VMs. This means that software running inside the VM can't interfere with the host or other VMs.

2. **Multiple Operating Systems:** You can run multiple VMs with different operating systems on a single host machine. For example, you could run a Linux VM on a Windows host, or vice versa.

3. **Resource Allocation:** VMs use a portion of the physical resources (like CPU, memory, and storage) of the host machine. These resources are managed by a piece of software called a hypervisor.

4. **Use Cases:** VMs are used for many purposes, including testing and development, running old software that requires a specific operating system, server consolidation, and more.

5. **Snapshot and Cloning:** VMs can be easily backed up and cloned. You can also take a snapshot of a VM to save the current state, which can be restored later.

Virtual machines provide a way to run software in a controlled, isolated environment, which can be useful in many scenarios.

A Virtual Machine (VM) is like a computer within a computer.

Imagine you have a Windows computer, but you want to use a program that only works on Linux. Instead of buying a new computer, you can use a VM.

A VM is a program that you run on your Windows computer, and inside that program, it looks and acts like you're using a Linux computer. You can install the Linux-only program in this "virtual" Linux computer and use it there.

So, a VM lets you run a different computer inside your existing computer. It's like having two (or more) computers in one!

-SSH Server & OpenSSH server

SSH (Secure Shell) is a protocol used to securely connect to a remote server. It's commonly used for managing servers, remote file transfers, and more.

OpenSSH is a free, open-source implementation of the SSH protocol. It's the most widely used SSH implementation and comes pre-installed on many Linux distributions.

When you select the "SSH Server" option during the Debian installation process, it installs the OpenSSH server. This allows other computers to make secure SSH connections to your Debian system.

Once the OpenSSH server is installed, you can start, stop, and manage it using the system's service management commands. You can also configure it by editing the SSH configuration file, typically located at `/etc/ssh/sshd_config`.

Linus kernel & Debian distribution

Linux is an operating system kernel. The kernel is the core part of an operating system that interacts directly with the hardware of the computer. It provides services like process management, memory management, device management, system calls, and security.

Debian, on the other hand, is a distribution (often shortened to "distro") of Linux. A Linux distribution is a complete operating system built on top of the Linux kernel. It includes the kernel, a package management system, libraries, tools, a graphical user interface (usually with a desktop environment), and pre-installed applications.

So, when you're using Debian, you're using the Linux kernel as part of a complete operating system that includes many other components. Other examples of Linux distributions include Ubuntu, Fedora, and CentOS. Each distribution has its own set of features, design choices, included software, and management tools, but all are built around the Linux kernel.

To implement the Born2beroot project at 42 school, you should have a good understanding of the following concepts:

1. Linux System Administration: This includes understanding how to install and configure a Linux system, manage users and groups, set permissions, and install and manage software.

2. Virtualization: You should understand how to create and manage virtual machines using software like VirtualBox or VMware.

3. Network configuration: This includes setting up IP addresses, configuring network interfaces, and understanding basic network troubleshooting.

4. Shell scripting: You'll need to know how to write scripts to automate tasks in the Linux shell.
5. Security: This includes understanding how to set up firewalls, manage user permissions, and secure a system against common threats.
6. Understanding of LVM (Logical Volume Management): This is a method of allocating space on mass-storage devices that is more flexible than conventional partitioning schemes.

7. Understanding of sudo, its configuration and usage.
8. Understanding of SELinux or AppArmor for mandatory access control security systems.
9. Familiarity with monitoring tools like netdata, nagios, etc.

10. Familiarity with Debian-based and RedHat-based Linux distributions.

Remember, the Born2beroot project is designed to be challenging and to push your understanding of Linux system administration. Don't be discouraged if you find it difficult at first. Keep learning and practicing, and you'll get there.

—-

1. Authentication and Access Control:

How do you configure user accounts and manage access control in Born2beroot?
- Create new_user: sudo adduser new_user
- Create group: sudo addgroup new_group
- Assign new user to group: sudo

Can you explain the process of setting up secure SSH access to a Born2beroot server?

2. Firewall Configuration:

Describe the steps involved in configuring a firewall using Born2beroot.
How do you ensure that only necessary ports are open while maintaining a secure server environment?
Package Management:

How do you use package management tools like apt or yum in Born2beroot to install, update, or remove software packages?
What precautions do you take when installing third-party software on a Born2beroot server?
System Hardening:

What are some common security measures you implement to harden a Born2beroot system?

Can you explain the process of disabling unnecessary services and reducing the attack surface of a server?
Monitoring and Logging:

How do you set up monitoring tools to track system performance in Born2beroot?
What log files do you regularly check for security and system health information?
Backup and Recovery:

Describe your approach to implementing regular backups for a Born2beroot server.
In the event of a system failure, how do you ensure a quick and reliable recovery using Born2beroot?
Kernel and System Updates:

How do you manage kernel updates in a Born2beroot environment?
What considerations do you take into account before applying system updates to avoid disruptions?
Security Auditing:

How do you conduct security audits on a Born2beroot server?
Can you provide examples of tools or commands you use to identify potential vulnerabilities?
User Permissions and Groups:

Explain how you manage user permissions and groups within Born2beroot.
What steps do you take to ensure the principle of least privilege for users?
Troubleshooting:

If a Born2beroot server experiences performance issues, what troubleshooting steps would you take to identify and resolve the problem?
How do you handle security incidents or breaches in a Born2beroot environment?