M A S T E R T H E S I S

Integrating Linux Hosts into Active Directory Environments - A Threat to a Company's Security Posture

submitted by:

Bright Jiwueze

(student number: 200088)

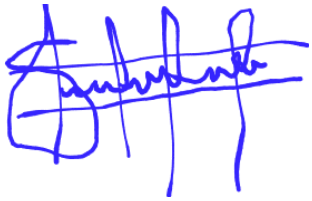
submitted on: August 6, 2024

Examiner: Prof. Dr. Max Moser

Supervisor: Prof. Dr. Max Moser

Statutory Declaration

I hereby declare that this work has been prepared without the help of third parties and only with the indicated sources and utilities. All excerpts used have been marked. This work has not yet been submitted to any examination authority in the same or a similar form.



Bright Jiwueze

München, August 6, 2024

Abstract

Integrating Linux hosts into Active Directory (AD) environments has become a common practice for companies that want a streamlined process in their IT operations and enhanced interoperability between several operating systems within the organization. While this trend has significant advantages regarding uncomplicated administrative tasks, centralized user management, and combined authentication, it also introduces several security threats that can destabilize a company's overall security posture. This master thesis inspects the vulnerabilities and problems of integrating Linux hosts into Active Directory environments. Key risks include increased attack surface, reconnaissance of the domain controller through the Linux host, AD service enumeration and exploitation using malicious payloads and packets, and potential exploitation of AD credentials. In addition, the difficulty in securing hybrid environments can give rise to several errors, misconfigurations, and omissions, further compromising the company's security posture. The master thesis also investigates best practices and mitigation strategies that could help to address these risks, such as encrypting all communications between the Linux and AD using TSL/SSL, reducing ticket lifetime to limit the ability of ticket reuse, employing robust access controls, regular monitoring, implementing advanced authentication mechanisms and Blocking unneeded traffic, ports, and scans, that can be used to send payloads from the Linux host to the AD Domain. By understanding and mitigating these security challenges, companies can better secure their AD-integrated Linux hosts and maintain a robust security posture.

Description of Task and Structure

The task is to identify the current threats integrating Linux hosts into Active Directory technology can pose to a company's security posture, then discuss how attackers can leverage those threats to circumvent administrative and security features already existing in the company. Firstly, a list of literature that had already been written about Active Directory basics, Linux basics, and their threats and security measures was reviewed, and findings were listed. We also got answers to questionnaires and surveys from companies that integrated these technologies, and we tried to learn about the current defensive mechanisms implemented on the integrated technologies. For data protection reasons, we did not include these companies' names or write anything specific about them. However, we used statistics to generalize some important and recent security mechanisms implemented on these technologies before and after integration. Other important findings revealed in this document are new threat and attack vectors built on top of the existing security features we found earlier. Finally, we discussed mitigations for these newly discovered attack vectors.

The structure of this master thesis relates to the discovery of new threat vector(s) that result from Linux and Active Directory integration. To achieve this, we discussed the basics of Active Directory and Linux, as obtained from our literature review. We did this to understand their individual roles and why most companies integrate them. Still, in the literature review, we discussed threats as a sub-topic and pointed out some current threats these technologies face today. We further emphasized attackers, who they are, and what they want. We also involved the cyber kill chain and MITRE ATT&CK frameworks in emulating how these attackers perform their tasks and how they can use attack vectors (threats) to circumvent existing defense mechanisms. Furthermore, through more research and questionnaires, we gathered information about the current security mechanisms implemented on these technologies, both in the separate and integration phases, and we prioritized these findings based on statistics. Subsequently, to achieve a result in this master thesis, we had a lab setup in a proxmox environment within the IP subnet 192.168.1.0/24 and a default gateway of 192.168.1.1. The reason is to provide a live demonstration of the complete integration process to give the readers a glimpse of how this integration works. From the laboratory experiment, we discovered new threats that integrated Linux can pose to our Active Directory environment. These new threats were discovered from an attacker's perspective. In addition, using a threat evaluation process, we elaborated on their likelihood of occurrence and impacts. Finally, we suggested additional administrative and security implementations.

Tasks:

- Literature work and questionnaires.
- Lab setup to demonstrate the complete integration process.
- Adversary emulation and Discovery of new attack vectors.
- Threat evaluation on the newly discovered attack vectors.
- Analyzing the obtained results and recommending a way forward.

Table of contents

List of Figures

List of Tables

List of abbreviations

AD	Active Directory
DC	Domain Controller
IP	Internet Protocol
DNS	Domain Name System
SSH	Secure Shell
POSIX	Portable Operating System Interface
GNU	GNU’s Not Unix
GCC	GNU Compiler Collection
GDB	GNU DeBugger
DPKG	Debian Package Manager
SQL	Structural Querry Language
JSON	JavaScript Object Notation
OWASP	Open Web Application Security Project
HTTP	HyperText Transfer Protocol
TGT	Ticket Granting Ticket

1. Introduction

1.1. AD-integrated Linux hosts vs. Cyber Defense

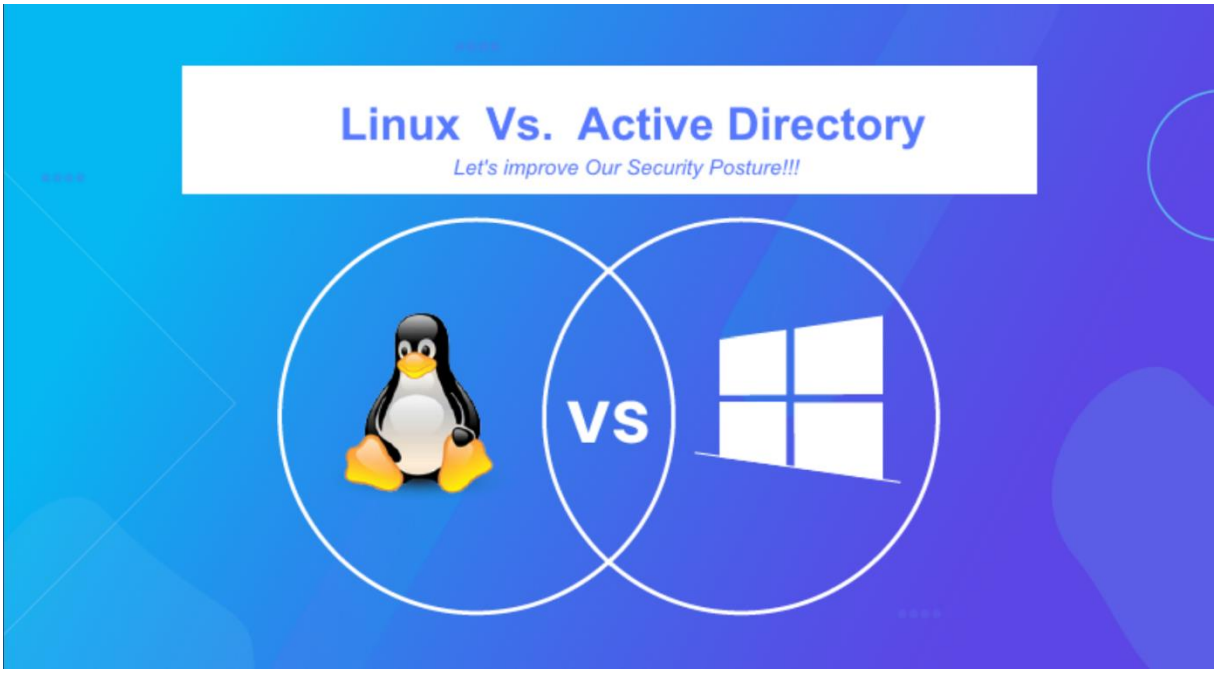


Figure 1: Linux and Active Directory

Windows Active Directory (AD) and Linux are technologies that have proven their importance in today's world. The underlying features of these technologies made them sophisticated enough in the sense that they could be used to manage the day-to-day running of a business. Based on statistical research, which we will also discuss later in the next chapter, there has been a speed increment in the usage of either each or, in many cases, both technologies. This proves that in the coming years, they will dominate the industry. These reasons make these technologies stand the test of time. Hence, we chose to make a detailed analysis of both from a cybersecurity perspective.

Furthermore, the best part of cyber defense is not just sitting opposite the screen waiting for an attack to occur; the implementation of active defense is key to building an excellent cyber-security posture. This means being ahead of the attackers and discovering vulnerabilities before the attackers' exploitation. Based on these facts, this document will help security engineers and penetration testers understand the present and future threats integrating both technologies can pose in their environment, encouraging them to be alert. This document would not only expose but also proffer some mitigation strategies that could be implemented for a robust usage of these technologies.

The major questions that would wrap around our curiosity as we go deeper into this research work include:
Why do organizations that use Linux hosts always see integrating them into their AD

environment as a better option?
Secondly, attack vectors are specific to individual technologies based on their development and design. From the view of a hacker or penetration tester, what types of threats do we think companies face after integrating these technologies?
Thirdly, what hardening strategies could be implemented after integration to ensure robust and secure usage now and in the future?

1.2. Securing AD-integrated Linux hosts to maintain a robust security posture

Today's world has produced many sophisticated hackers who, by any chance, can detect cyber threats and, through them, can find their way into a company's network. This is why there is a high rate of cybercrime in today's world. To reduce the effect of this cybercrime, technologies used for the day-to-day functions in enterprises should be secured so that hackers, no matter how sophisticated they are, would find it difficult at any trial to have their way into the company's network. This document will x-ray Active Directory and Linux integration; we will discuss the threats this can pose to a company's security posture. Our discoveries would give rise to various defensive mechanisms, thereby assuring law enforcement agencies and end users of well-secured technologies now and sometime in the future. This document would also provide a basis for future research on this topic.

This document will emphasize the underlying features, usage, and characteristics of Active Directory and Linux technologies. Our interest and curiosity are limited, primarily within the umbrella of cybersecurity. Therefore, we will view both technologies from an offensive and defensive point of view. Furthermore, the procedure we will leverage in this work to achieve our desired result is to perform a live integration of both technologies in our laboratory environment. During this integration, our concentration would be mostly on the attacker's perspective to figure out unavoidable misconfigurations that can help attackers circumvent the existing security features to get an initial foothold or escalate privileges on the Active Directory environment through the Linux host. These circumvention approaches will form our newly discovered attack vectors, and on them, we will base the result of our hypothesis.

In the next chapter, we will examine the background of our work and try to understand some important aspects and underlying features of these technologies that make them relevant today. While we examine the cybersecurity perspective deeply, we will also examine the contributions made by other authors in the direction of our work and understand the limits of their works before performing our evaluation to achieve our results.

2. Background and Related Work

2.1. Usage of These Technologies

Evaluating this important topic without first discussing some reasons that made this research work stand the test of time might raise some questions about the quality and timeliness of this document. Let us first discuss the relevancy based on usage.

Before we move further, let us first talk about the quality. This document had a broader look into Linux hosts and Active Directory environments from a technical point of view, making it much easier for the readers, to some extent, to have a great insight into the underlying features that make up these technologies. Generally, the information contained in this document solved two essential problems. As we all know, these are two different technologies produced by different people for different purposes and with different underlying features and attack surfaces; therefore, solving the problem of integrating both would also, to some greater extent, solve the problem of using these technologies differently.

The author looked deeply into today's industry and noticed that most systems and applications used in industries today are hosted on Linux servers. Various companies also use Linux OS daily, especially those requiring writing and executing codes and scripts. Also, at the time of writing this document, the market share of Microsoft Active Directory was about 58,836 customers (6sense, 2024). Let us note that since the inception of these technologies by Linux and Microsoft, respectively, their patronage has never decreased; the statistical usage keeps increasing in geometric progression. Companies continually leverage Linux hosts and Active Directors to perform tasks and manage the day-to-day running of their businesses. The reason is that the underlying features of Active Directory technology as a sophisticated and proprietary directory service make it capable of bringing together all the tools and technologies used in the company. In most cases, authorized users use single-sign-on authentication to access all these technologies using the password and username associated with their AD accounts. Most companies today use Linux in their day-to-day activities, whether as a user-specific OS or server. In most cases, when Linux is used as a web server, admins can connect to it using their AD credentials to perform system configurations. For this reason, these companies try to integrate the Linux hosts into their Active Directory environments to manage all authentications from one specific endpoint.

Furthermore, as we move further, let us not easily forget that these different environments have different attack vectors before integration. In the world of human beings, you can leverage your attack vectors through the children from a particular family to attack their parent(s); likewise, you can also attack the same children via their parent(s). This explains that an attacker can enter through the parent or the child domain.

Though the scope of this research work is limited to only Linux hosts integrated into Active Directory environments, the above analogies indicate the need and authenticity of this topic, thereby providing a clearer view of the quality of this document. Also, just as bees are attracted to honey, these technologies' outrageous growth and dominating features make them a point

of attraction for hackers. We are judging from the facts that today's technology has produced many sophisticated hackers who are well-trained and sponsored on how to gain an initial foothold or escalated privileges on these technologies. To this end, we can justify the timeliness of this research work as being authentic.

2.2. In-depth Description of the Technologies

Until now, we have only succeeded in understanding the quality and timeliness of this research work based on usage statistics. At this phase, we will switch to technical research to understand the logical description of these technologies.

Just like a human being having different organs and different body parts functioning together internally to help humans achieve their daily tasks, we will explore laterally these technologies to have a full grip on their underlying parts and pieces, describing them according to their specific functions and finally discuss how these parts and pieces network together to achieve a primary function. In this section, we will concentrate more on important features that hackers target, so we will not go deeper into learning how to use these technologies after deployment because it is out of the scope of this research work. The descriptions of these two technologies will be done independently.

2.2.1. Active Directory Basics and Important Features

Like every other directory service, Active Directory, which can also be seen as a network operating system, is a software service built by Microsoft that has existed for over a decade. Its usefulness is in managing the overall information of an enterprise efficiently. It works like a lightweight directory access protocol (LDAP), which is a protocol designed to maintain access to a directory service via the network. Active Directory management is done on a centralized server and can be accessed globally via the Internet. It stores an organization's critical information, like databases, users and groups, computers, applications and services, operating systems, and other infrastructures that can be used to run the day-to-day activities of a business. All these elements are stored inside the data store, also known as directory domains. They are being stored as objects inside the data store, which grants users and administrators access to these stored objects via the internet. The overall management of an Active Directory is done by the system administrator(s), who manages users' entry and exit from the Active Directory domain by providing authentication mechanisms that allow users to access the domain when the provided credentials are true for a particular user account. Administrators also give permissions to access information within the domain; these are given based on individuals or groups and are read, write, and full control permissions.

Like every other software designed to run on Windows machines, Active Directory has no exceptions, except for the fact that it is seen as a **domain** and is installed on a **Domain Controller** hosted on a Windows server.

2.2.2. Domain vs. Domain Controller

A domain is a logical environment that houses related network entities, e.g., users, computers, applications, resources, etc. It groups these entities into manageable objects within a secured environment and ensures that authenticated users can only have access to information they are authorized to. A domain must have a name known as a domain name and an IP address given to it by the Domain Controller, which is made publicly known to the users or members of the enterprise. With the IP or domain name, end users can access the Active Directory from a web browser. You can liken an Active Directory domain to this image below.

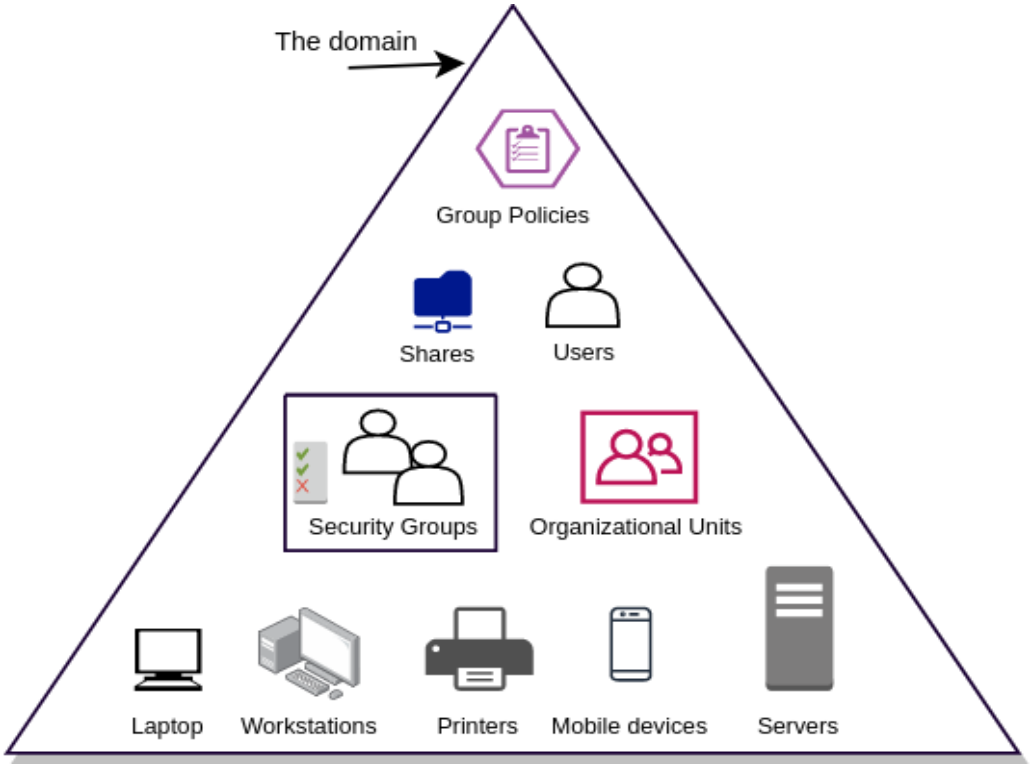


Figure 2: AD domain with entities connected within a network (Perishable, 2023)

Once a user is authenticated into the domain via some authentication mechanisms, they can access any entities based on permissions, enterprise rules, or security control.

Domain Controller: This is the brain server of the Active Directory environment. It is the local server where the Active Directory domain is stored. It gives existence to the domain and facilitates users' experiences and the availability of resources. Furthermore, this is the environment where the Active Directory's IP address and DNS server are configured before deployment. The Domain Controller has a database that stores user credentials and only grants access to the domain when a user's credentials match the stored credentials. Only the system administrators have shell access and full control permission to this region via SSH to affect some security mechanisms, configurations, access rights, and permissions based on the enterprise's security controls.

2.2.3. Active Directory Installation

Firstly, it is important to understand that, as was specified earlier, Active Directory, as a proprietary Microsoft-initiated service, can only run on a Windows server. Before installing the Active Directory using the installation wizard, a Windows server must be configured to function as a Domain Controller (DC). To achieve this, a Windows server must be downloaded and installed on-premise or in a cloud environment. For a life demonstration in this work, we used Windows Server 2022, which we downloaded and installed on a proxmox environment. To make it a Domain Controller inside the Windows server, we changed the hostname of the local server to reflect the name of an existing Domain Controller; in our case, we use BJ-DC1. We configured the IP address as a static IP based on our network ID, subnet, and default gateway. In some cases, especially in an enterprise environment, we might choose to have a separate DNS server and direct DNS traffic to it using a firewall or other device. In this demonstration, the DNS was installed in the Domain Controller and pointed to the IP address of our DC. After the host and IP configurations, we clicked on the server manager dashboard, then navigated to the 'add roles and features' directory and selected features to install on the local server BJ-DC1 that we configured. In our case, we selected an Active Directory domain service and a DNS service, accepted all the roles attached to them, and finally performed the full installation. Then, to promote the window server to a Domain Controller, we created a forest alongside the forest root domain that we named subaac.com (see a description of the forest in the next section). We also created the forest and domain functional level, determining the capabilities and operating system on which the Domain Controller can function. Then set a password for Active Directory service restore mode in case you need to perform an Active Directory recovery in the future. After the installation, we rebooted the machine to complete other installation features. Of course, because of the scope of this section, we did not go deeper in our demonstration. It is also important to know that we can create more than one Domain Controller to manage our Active Directory domain. We can synchronize them together so that any administrative update can be made to the Active Directory domain via any Domain controller. This is because when one Domain Controller is down, others will continue to provide functions. It can also solve the purpose of load balancing in the case of a larger network. As stated earlier, we installed the DNS settings on the Domain Controller and pointed the IP address to the Domain Controller. We configured the reverse lookup zone and added our Domain Controller pointer record.

2.2.4. Active Directory Authentication

Any access to the AD environment must begin with the Kerberos authentication process. The user will send an Authentication Server Request (AS-REQ) message to the Domain Controller (DC). This will automatically encrypt the timestamp of that message request with the user's hashed password and username. the Domain Controller receives the request and attempts to decrypt it with its record of the user's password hash stored in the **ntds.dit** file. If this is successful, the Domain Controller will send back an Authentication Server Response (AS-REP) message containing the user's session key encrypted by the user's password and a Ticket Granting Ticket (TGT). The user will use the TGT to access the requested domain resources

inside the domain.

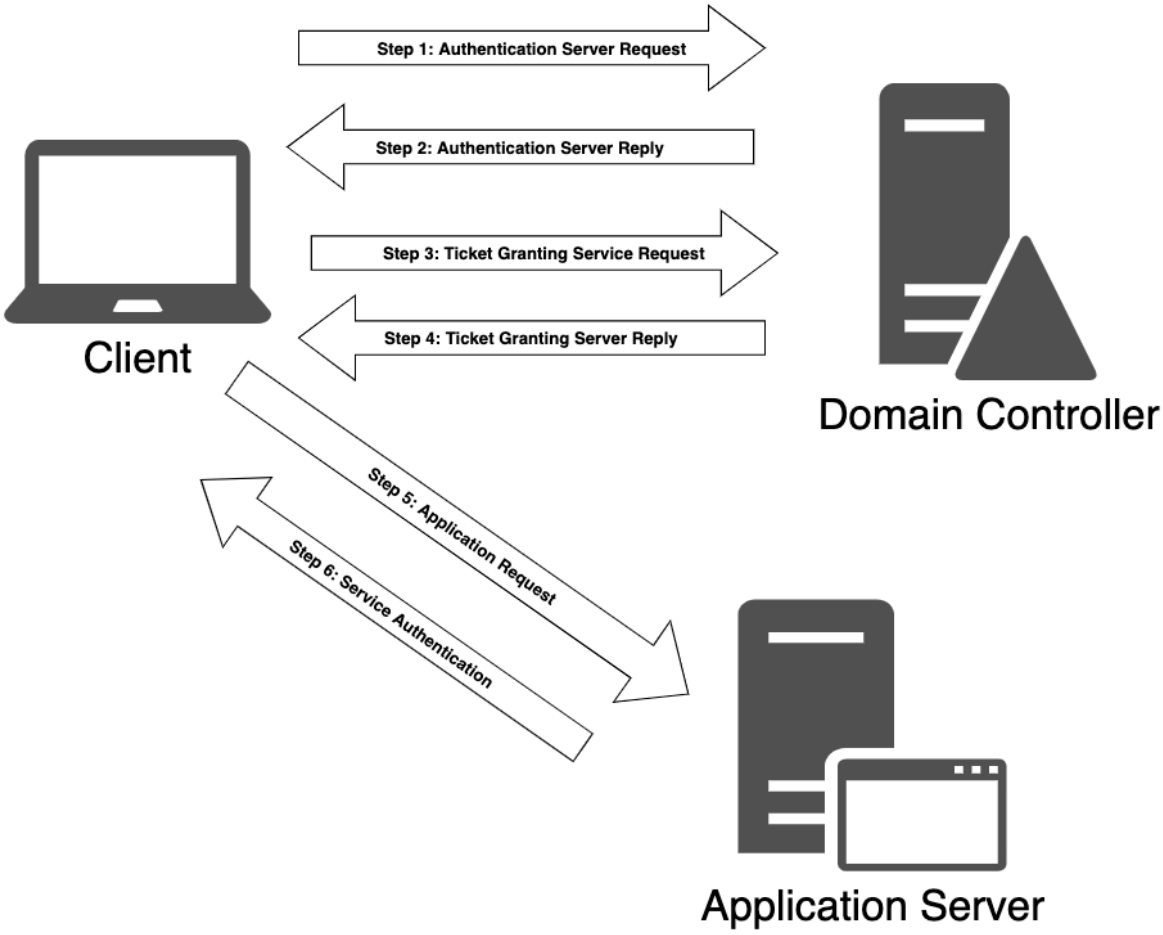


Figure 3: Active Directory Client Kerberos Authentication (Offsec, 2023)

2.2.5. Forest vs Organizational Units

A **forest** is a root domain that serves as a container for all the domain trees (child domains) created in the Active Directory. The forest root domain name is the root domain name the DNS server uses to direct traffic to the Active Directory domain. Other subdomains known as trees or child domains derive their names from the root domain name, e.g., if a domain name is *subaac.com*, other child domains can get their names like *ubu-bc.subaac.com*. These trees or child domains can be managed differently by other administrators in the case of a large enterprise scattered at different locations, while the enterprise system administrators manage the root domain. The database file **Ntds.dit** provides the physical storage of all object accounts in a single forest.

Organizational Units: This provides a means of organizing or grouping objects inside the domain. It helps administrators issue administrative rights and includes group policies for groups of objects. The organizational units can be used to mirror an organization's business functions and structures. Inside the Active Directory, objects can be grouped based on specific functions, categories or a specific set of users and groups, and accounts can be created on those grouped objects, so access to it must be authenticated and authorized based on some mechanisms and roles. For example, a supermarket implementing an Active Directory to run its business; in that Active Directory environment, the objects a human resource manager can access are different from those a cashier can access, and so are others. This is the concept of implementing organizational units inside the Active Directory environment. It helps to organize objects in the AD environment.

2.2.6. Importance of Active Directory

Finally, before we move to the next technology, let us highlight the importance of Active Directory according to Microsoft (Archiveddocs, 2014).

 Active Directory provides a central location for network administration and delegation of administrative authority. You have access to objects representing all network users, devices, and resources and the ability to group objects for ease of management and application of security and Group Policy.

 Information security and single sign-on for user access to network resources. Tight integration with security eliminates costly tracking of accounts for authentication and authorization between systems. A single username and password combination can identify each network user, and this identity follows the user throughout the network.

 Active Directory is a scalable technology that includes one or more domains, each with one or more Domain Controllers. This enables you to scale the directory to meet any network requirements.

 Active Directory is flexible and provides global searching. Users and administrators can use desktop tools to search AD DS. By default, searches are directed to the global catalog, which offers forest-wide search capabilities.

 Active Directory provides storage for application data. It provides a central location to store shared data between applications and with applications that need to distribute their data across entire Windows networks.

 Active Directory systematically synchronizes directory updates. Updates are distributed throughout the network through secure and cost-efficient replication between Domain Controllers.

2.2.7. Linux Basics and Important Features

Linux is an operating system created by Linus Torvalds in the 1990s. It is derived from the Unix operating system and leverages the POSIX standard, which is the standard Unix was built on. POSIX is a standard for maintaining portability among operating systems. All operating systems built based on the POSIX standard behave alike and are compatible, meaning that any tools used on one can also be used on others.

Unlike the earlier technology discussed, Linux is not a proprietary service; it is not restricted to commercial purposes, is not owned or managed by a specific company, is not complicated, and does not need specialized training to be used. Linux is developed for personal computers and can always be modified and redistributed. It is an open-source software application written in the C programming language. The C compiler is included in it by default, which means any program written in C can run on a Linux operating system. It is also important to note that the Linux system is compatible with the GNU software. Most tools produced by the GNU project can be used on Linux, which include Bash, GCC compiler, GDB debugger, core utils such as ls, cat and ch mod, find utils to search for files, font utils to convert fonts from one format to another or make new fonts, Emacs used for editing, octave to perform numerical computations, gnu SQL for relational database systems, etc. To install packages on Linux, you will need RetHeart package manager (RPM) or DPKG, though it depends on the Linux distribution, which we will discuss later in this chapter. However, generally, we use apt-get to install packages in any Linux distribution. Other Linux commands can be found on the Linux manual pages. You can use the *--help* option to learn more about a particular command.

Linux is like a software box that can be installed on hardware as the underlying OS or hosted inside a network as a server. It can be downloaded as a VMware file, a disk image (.ISO) file, or a Docker image file. For either of them, the configurations and utilities are the same, except that installing it in hardware could be for enterprise, personal, or educational purposes while hosting it as a server to be accessed via the network is mainly and solely for enterprise uses. Linux already has existing pre-installed features and can be accessed via root. You can leverage the existing features alongside the command lines to recreate the Linux box for your specific use. The structure of Linux starts with the parent directory, known as ROOT, which is only available for privileged users. The root directory contains all the directories and files that hold Linux configurations. Companies also store all their important information in this parent directory so that only privileged users, such as administrators, can access it. Once an account is created during installation, Linux will automatically add the account as a privileged user. This user account is automatically assigned to the sudoers group and can perform all privileged functions as a root user using **sudo**. We will discuss **sudo** in our next chapter. Understand that the root password can be changed if the admin wants access as root and not as a normal user with specific privileges.

2.2.8. Linux Files and Processes

When we investigate a Linux environment, either as a privileged or unprivileged user, we will understand that the instances within the environment are categorized into two: files and processes. This means that once you have access to a Linux environment, you are there to communicate with either a file (directory) or a process. It is also important to know that applications, servers, and programs running on the Linux platform, either in the background or foreground, are traditionally known as processes. All Linux processes run independently; to capture all processes via the Linux terminal, type the command *ps aux*.

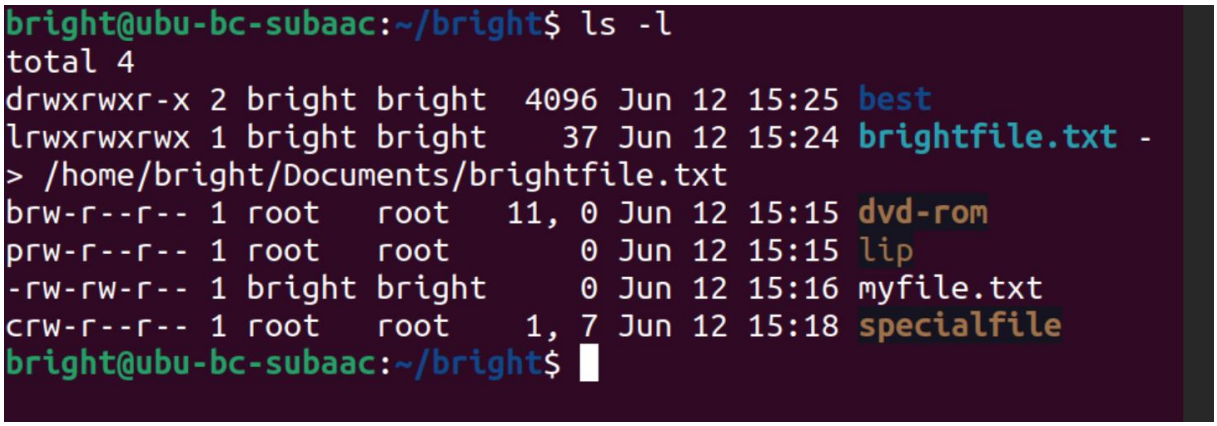


Figure 4: Different types of files in a Linux environment.

Symbols	Meaning
D	Directory
L	Link
B	Block device
P	Named Pipe
-	Regular File
C	Special file

Table 1: Linux files and their respective symbols

Every file, as well as every element of a Linux system, is surrounded by user and group permissions following three basic properties: *read* permission (symbolized by **r**), *write* permission (symbolized by **w**), and *execute* permission (symbolized by **x**). Files or directories have distinctive permissions for three categories of users, which include the owner (the owner or creator of the file), *the* owner group (all members that belong to the owner's group), *and* the other group (other users that are not owners and do not belong to the owner's group).

Looking at the screenshot in Figure 4, we noticed that the files, irrespective of their types, are accompanied by permissions. As stated in the paragraph above, these permissions are separated into three categories. Each of the permissions (rwx) allows a user or collection of users to perform specific functions on the files within the environment. However, it depends on the type of file the user is working on, which we will describe below (Offsec, 2023).

For regular files and other file types, **r** permits reading the file's contents, **w** permits changing the contents, and **x** permits the file to be run as executable. For directories, they are handled differently from other types of Linux files. **r** (read access) gives the right to retrieve contents. **w** (write access) permits creating or deleting files in the directory. Finally, **x** (execute access) permits entering the directory to access its contents (it can be done using the cd command). On Linux, the **chmod** command can modify these permissions to suit who should have access and perform functions in a particular file or directory. In Figure 4 above, the regular file **myfile.txt** has **rw** permission for the owner **bright**, meaning the owner can only modify and read the file but cannot execute the file. The next category of permissions is for the members of the file owner's group; they also have **rw** to perform the same function as the owner. The last category of permission is for any other user with access to the environment; they have only **r** permission, which means they can only read the file but cannot do anything on it. The change mode command **chmod**, alongside elevated privileges, can include or remove any permissions to suit specific or enterprise needs. Recursive permission can also work alongside **chmod**; this is a special case where all files within a specific directory can derive their permissions from their parent folder.

2.2.9. The Linux Command Line Interface and Command Execution Process

Unlike Windows, which is more user-friendly because a user is presented with a nice Graphical User Interface (GUI) after successful authentication. Linux mostly depends on strings of executable commands to perform its functions. These strings of commands are either shell built-in, installed alongside the operating system, or installed specifically by the user. These commands are executed on the command line interface (Linux terminal). This is an interface or terminal where a user can input a series of Linux commands and get a result if the user has permission to run such a command and the Linux kernel is configured to respond to the command. The terminal is the interface that directly communicates with the end user. It receives input from the user as a command and passes it to the shell. The Linux shell is a program that interprets user commands entered through the terminal. It interprets the command and passes it to the kernel for execution. The Linux shell is categorized into many types, but the most frequently used one is the Bourne Shell (sh). The Bourne Shell is the default shell for almost all Linux distributions; another important Linux shell is the Bourne-Again Shell (bash), which is like a higher version of the Bourne shell and interprets single and batch commands. We have many other Linux shells that can be found in different Linux distributions.

When a Linux User enters a command on the terminal, the shell will check an ordered list of paths to check if the command has an executable binary and can be executed by the user on the terminal. This ordered list of paths is contained in an environmental variable known as

$PATH. Every user in Linux that has a home directory must have this variable to check for existing executable commands. This PATH variable uses two important directories where Linux executable binaries commands are stored. These directories are the /bin and the /sbin directories. While the /sbin stores the executable binaries that can only be executed by a privileged user, the /bin directory stores the executable binaries that can be executed by a normal user. When a user runs a command, the PATH variable will check the /bin for a normal user and both /sbin and /bin for a privileged user, and if the PATH variable searches both directories and cannot find the executable binary for the user's entered command, this means the user needs to install the command through apt or apt-get repositories. However, if the executable binary for that command is found, then the shell will make a system call to the kernel. The PATH variable can be manipulated for any user to allow or deny some executable commands from the user on the shell; this means that through the PATH variable, a privileged user can be restricted from running some privileged commands, and an unprivileged user can be allowed a privileged command.

```
bright@ubu-bc-subaac:~$ cd /usr/bin
bright@ubu-bc-subaac:/usr/bin$ ls |grep ifconfig
bright@ubu-bc-subaac:/usr/bin$ cd /usr/sbin
bright@ubu-bc-subaac:/usr/sbin$ ls |grep ifconfig
ifconfig
bright@ubu-bc-subaac:/usr/sbin$ cd
bright@ubu-bc-subaac:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games:/usr/local/games:/snap/bin:/snap/bin
bright@ubu-bc-subaac:~$ ifconfig
ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.120  netmask 255.255.255.0  broadcast 1
92.168.1.255
        inet6 fe80::ae08:71d9:3d6a:4167  prefixlen 64  scopeid
 0x20<link>
        ether 2e:4a:8d:5b:8f:7d  txqueuelen 1000  (Ethernet)
        RX packets 612881  bytes 498499796 (498.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 115515  bytes 11061804 (11.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisio
ns 0
```

Figure 5: Checking the PATH variable for command execution

In Figure 5 above, we took an example of a network administration command 'ifconfig'. We first navigated to the /usr/bin directory to search for the command's executable binary. However, our search command returned nothing because it is a privileged command and cannot be found in that directory. However, when we navigated to the /usr/sbin directory, we found it.

Now to run the command, we checked the PATH variable for the user and noticed that /usr/sbin is included in the PATH for that user; we executed the ifconfig command and got a successful output.

Next, we will remove the /usr/sbin directory from this user's PATH and try the ifconfig command again to see whether we get a different result.

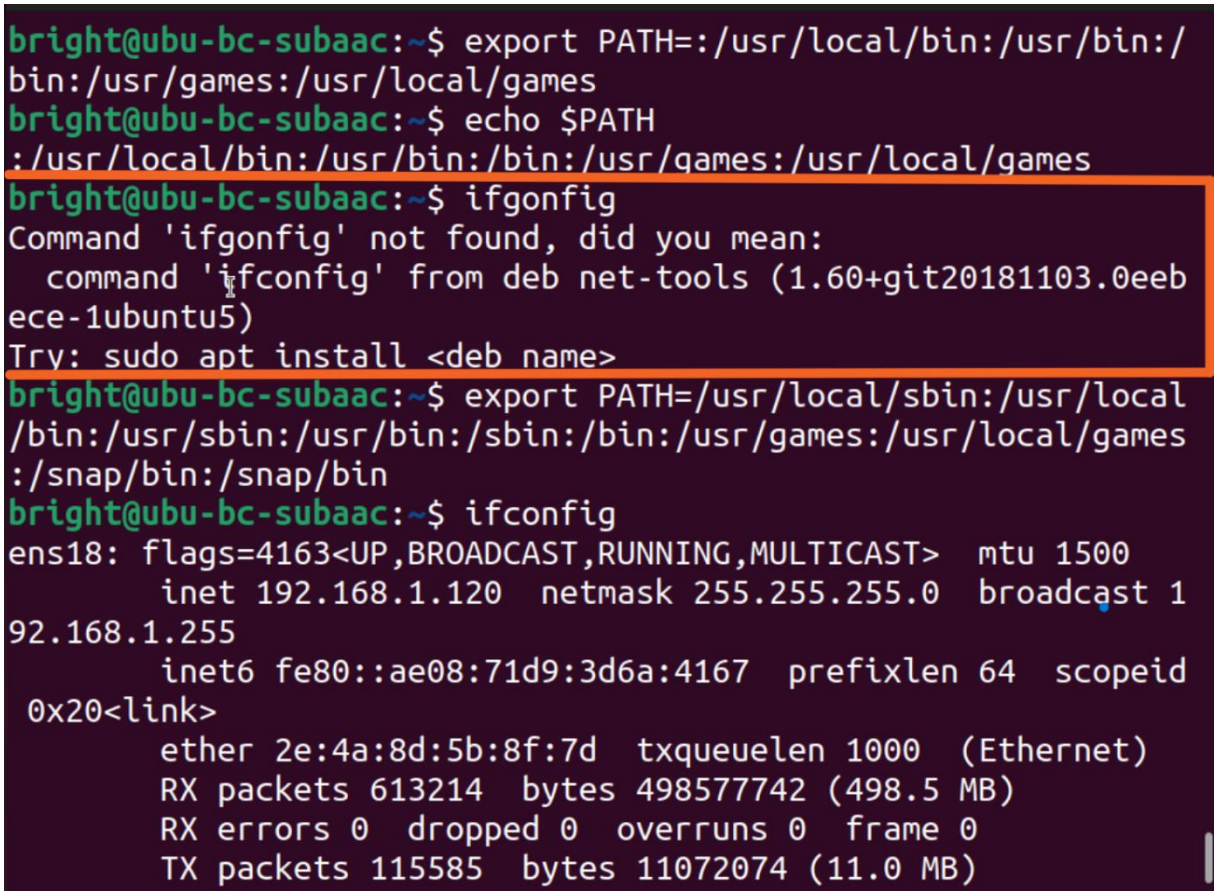


Figure 6: Manipulating the PATH variable for a user

In Figure 6 above, we used the export command to overwrite the list of paths in the user's PATH variable and remove any trace of /sbin from it. This made the user unable to run the 'ifconfig' command because the shell could not find the executable binary for that command in the /bin directory. Everything worked as usual after we restored the PATH variable to the default.

The description we made in this section is to give us a closer overview of how Linux commands are transmitted from the terminal down to the kernel space. In the later sections, we will discuss the Linux kernel.

2.2.10. Linux Distributions

While we progress in this section to discuss some important Linux distributions, it is crucial to understand that all distributions play the role of a stable server and a good desktop client. All Linux distributions contain the same set of basic packages, except for the fact that special third-party software is added for specific functions, which gives rise to different kinds of Linux distributions.

A typical Linux distribution is an operating system made from a collection of software and files that runs on top of a Linux kernel and comprises an init system, GNU tools and libraries, documentation, and other Linux features. All Linux distributions are built based on user-specific needs.

The most popular Linux distributions as of the time of this documentation are Debian, Ubuntu, Kali, CentOS, Fedora, Mint, Red Heart, Manjaro, Solus, Elementary OS, etc. All these Linux distributions listed have the same structural basics but different third-party features to fit user or enterprise needs.

2.2.11. Linux kernel

Technically, the kernel, which is the heart of the Linux operating system, is the intermediary system between the hardware and the operating system. It can work on both physical and virtual hardware to manage process execution and proper resource allocation, which includes sharing available resources like network connections, CPU time, disk space, and so on—among various system processes. It is important to know that programs, files, and processes cannot communicate directly with the hardware; they need a technology that can grant them the hardware resources they need to function properly, and because the kernel has unrestricted access to the hardware resources, it solves the problem of managing memories and scheduling processes for proper system functionalities.

To simplify the above explanation, we can understand the kernel's job via this diagram.

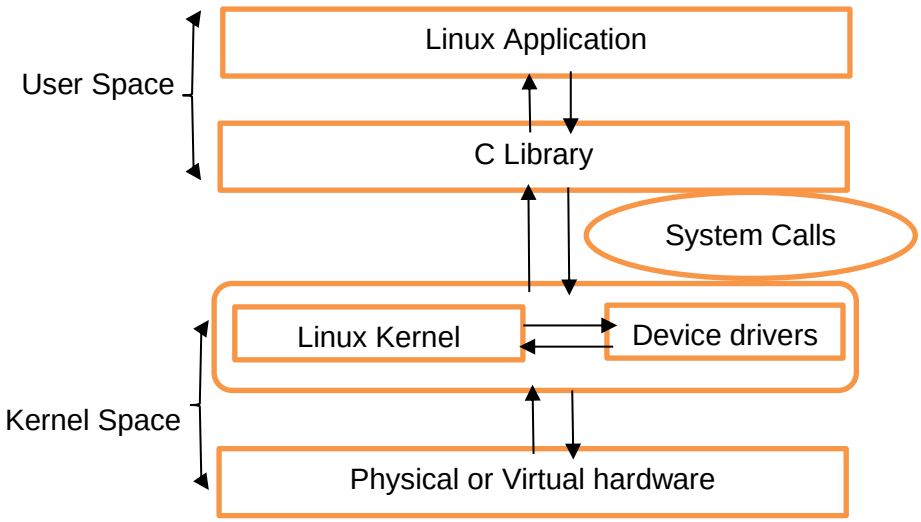


Figure 7: The Linux kernel responds to users' needs via a system call

This separation between the user space and the kernel space is to prevent user applications from directly accessing the system's critical resources. The user space can only interact with the kernel space via **system calls**. For example, when you type the Linux command ***ls*** in the user space, a system call (getdents()) will be made, which tells the kernel to list all the files that belong to the current directory. The kernel will respond to this system call as a user-authorized request, and then the result is executed in the user space. We will elaborate on this concept more from a security point of view in the next sections.

Before we conclude this section, we will highlight the importance of Linux technology.

- Linux is an open-source technology, meaning that you can modify, use, and distribute the source code. This encourages constant updating and customization to meet user-specific needs.
- Linux has nice security features. Based on our earlier discussions about this technology, we highlighted some of the important security features; we will also understand others as we advance further in this paper. Because of its security features, most companies use Linux as servers and to manage other critical systems.
- Linux technology's open-source nature makes it cost-effective. Individual organizations can obtain it freely and use it as they wish.
- Linux operating systems' underlying features make them compatible with a wide range of new and legacy hardware technologies.
- The community support from the large community of Linux users and distributors can offer advice and help extensively. This makes it easier to implement and use.

2.3. Related Work

At this stage, in this document, we will look around into some works of literature written by other authors that are relevant to the topic we are discussing. First, we will review what other authors have achieved regarding current cyber threats faced by these technologies. Also, we will involve some existing threat intelligence and adversary emulation frameworks to understand how attackers take advantage of threats and turn them into attack vectors. Furthermore, we will review Literature and questionnaires about Active Directory and Linux securities and prioritize them based on statistics. This will help us understand how organizations mitigate these threats to build a proper security posture.

To gain a more high-level understanding and bring the literature review section much closer to our topic, we will also investigate the integration explained by RedHat Enterprise to understand their perspective based on the ideas they are communicating. Finally, we will investigate properly to see whether other authors provided relevant points about Linux and Active Directory integration security.

The general information we will gather in this chapter will expose us to what other authors have done, which will give a better ground in this document for us to perform our experiment and generate our result not to disqualify what other authors have done but to expose us to some

other areas they did not touch. This will also guide us in understanding how wide the attack surface could be after integrating Linux hosts into Active Directory environments and how to implement more security measures, which will help us build a strong security posture in our enterprises.

2.3.1. Current Cyber Threats Faced by Linux and Active Directory

Before we proceed in this section, let us first understand the concept of cyber threat. Many authors have written about cyber threats. We will explain them broadly and then limit them to the topic of this work.

Cyber threats, also known as security flaws or potential attack vectors, are intentional or unintentional misconfigurations that can create attack vectors and engineer a successful cyber-attack. We can also say that they are tunnels through which unwanted and malicious traffic can go in and out of a network perimeter, disrupting business activities by tampering with critical data and assets that contribute to the growth of the business. They are mostly caused by vulnerabilities in a network. In some cases, these threats might be zero-days which means that they might not be known to the persons in charge of patching the flaws because when a malicious actor discovers an attack surface in your network before you, this will eventually pose a zero-day threat to security posture.

To Limit the scope of this section to our work, let us discuss the current cyber threats these technologies face. We will first emphasize Active Directory, then conclude with Linux.

The current cyber threats Active Directory technology is facing today:

Employee Neglect: Up to this moment in the world of technology, human beings e.g. employees, have always been major attack vectors used to get unauthorized access to an enterprise network. In most cases, it is either intentional or unintentional. Intentional is when an employee is convinced with certain promised or physical cash and is used to gain unauthorized access to a network. In most cases, some of these employees might be naive not to be properly aware of the consequences of their actions both to themselves and to the enterprise in general. In some cases, it might be because of an unfriendly working environment where the employees are not allowed to communicate their views, those employees might see the illegal promises as more profitable to them than the growth of the company. Secondly, unintentional is when an employee forgets to adhere to or meet up with some security policies of the company, e.g. clicking links and attachments attached to phishing emails, visiting unsecured websites, downloading unauthorized software, etc. All these can grant unwanted access to the company's network either to get an initial foothold or to increase privileges. Staff training, awareness, legal consequences, and nice working relationships and communications can subside threats of this kind.

Neglect To Third-Party Risks: Today, it is common for organizations to leverage outside sources known as third-party vendors. This method is broadly known as supply chain. Let us say a third-party vendor has a software application needed by **company A**, that could help

company A to respond to their customers' needs. This third-party vendor's security practices might not be strong, this can cause an attacker to gain access and install malware into the software. **Company A**, which is using Active Directory to manage and streamline all their company's processes, will buy the software and install it in their Active Directory environment. This common mistake will give the attacker access to **Company A**'s network. Paying attention to risks associated with most third-party tools can help minimize this kind of danger.

Unsecured Objects in The Active Directory Environment: In the Active Directory environment, objects can be found in the form of user accounts, group accounts, computers, databases, and servers found in the environment. Any of these entities, if not handled properly, can pose a threat to the Active Directory environment. Ranging from improper offboarding and deletion of users and group accounts to access violations on servers, computers, and databases can open a door for unauthorized users to find their way into the company's network.

Unsecured Programs and Scheduled Processes: Most companies only allow employees to install programs on their systems using administrative permissions. These programs after installations run in the system space **C:\Programs** not in the users' space **C:\Users**. It has been seen in most cases where users leverage programs or tools that might facilitate their jobs. Without proper knowledge, legacy software can be installed on the system; careless installation like this without proper research can pose a threat which allows any unauthorized access to escalate privileges. On the other hand, most admins schedule processes and forget to set proper permissions for those processes. To them, provided the process is working perfectly, the task has been done and marked closed. Processes like this with insecure write permissions can be used to create a privileged account that can grant unauthorized system-level access to the network.

Service Misconfigurations: Active Directory services are entities built in the AD environment that facilitate users' needs, while scheduled processes help with system needs, services are configured majorly for users' needs, and they can serve as intermediaries between the user and other objects in the environment. Most of these services have registries that receive and store users' credentials either as clear text or as hashed credentials because they are made to authenticate and grant users access to objects they are only allowed to access. This explanation entails that, once you are authenticated into an Active Directory environment as a user, other entities you are to access have services that also receive your credentials and, most times, store them in their cache register for subsequent access. When these service registers are misconfigured, they can leak users' credentials either as hashes or as clear tests, and if the user account is privileged, it can grant privileged access to the system.

AS-REP Roasting: This is a threat that exposes the hashed passwords of user accounts that have their Kerberos preauthentication disabled, this hashed password can be cracked offline when retrieved. Kerberos authentication was discussed earlier in this document, which explained how a user that requires access to resources in the AD environment can perform the Kerberos authentication process in other to acquire the Ticket Granting Ticket (TGT) to access

any resources in the AD environment.

However, this authentication should be enabled for every user by default, but for some reasons caused by omission or commission, administrators might disable preauthentication for a user; this could cause an attacker to request the user's authentication data from the Domain Controller. The Domain Controller would return an AS-REP message in response to the request. Since the user's password has been used to encrypt part of that message, the attacker can then perform an offline brute-force attack on the response.

This threat can be exploited via Kali Linux using **impacket-GetNPUsers,** and via Windows using **Rubeus.exe** tools. Microsoft documentation says administrators can use this Windows command to determine users with Kerberos preauthentication disabled '***Get-ADUSer -Filter { DoesNotRequirePreAuth -eq $true } -Properties DoesNotRequirePreAuth | select SamAccountName, DoesNotRequirePreAuth '***. Once detected, it can be enabled with the command: ***Get-ADUSer -Filter 'DoesNotRequirePreAuth -eq $true '*** **| Set-ADAccountControl -doesnotrequirepreauth $false** (Thameur Bourbita, 2023).

Kerberoasting: This is a Kerberos authentication threat associated with the Service Principal Name (SPN). Service Principal Name is the unique identifier of a particular service instance. Whenever a client wants to access a service provided by the SPN, the client will request a service ticket provided by the Domain Controller. The service was originally designed so that the application server can only decrypt it because it is encrypted with the password hash of the SPN. When requesting the service ticket from the Domain Controller, the Domain Controller does not check whether the user can access the service. The verifications are only done when connecting to the service itself. This indicates that if we know the SPN-provided service we want to target, we can request a service ticket for it from the Domain Controller, and since the service ticket is encrypted using the SPN's password hash, it means if we can be able to decrypt it using brute force or guessing, then we can have access to the cleartext password of the service account. This threat can be exploited via Kali Linux using **impacket-GetUserSPNs and** Windows using **Rubeus.exe** tools.

Now, **let us also discuss the current cyber threats Linux technology is facing today.**

Abusing System Daemons and Scheduled Processes: These are customized, or user-specific services scheduled to either start up at boot time or run at a specific time and interval. They can also be triggered by an activity such as adding a new user, installing a new application, etc. In most cases, system administrators ´rely on some of these processes to execute some ad-hoc tasks, and most of these processes run on privileged permissions and might expose some sensitive information during process enumeration. System administrators also schedule some jobs e.g., cron jobs to run some scripts at a scheduled time or interval, and they always forget about file permissions before hosting the script. A malicious actor can tamper with these scripts to get a shell as root.

Abusing SetUID Binaries: Setuid or Setgid permission entails that a user or group, not root and not in the sudoers group, can execute a command on a root file without having privileged

permission. Normally, Linux processes run with the UID or GID of the user that started them. Unless you are the user or group member of the user who started the process, you cannot have access to that process. Linux has a mechanism to circumvent this for easy workflow. A typical example of this is when normal users can change their passwords in the /etc/shadow file. In normal circumstances, this can be done as root or as a member of the sudoers group, but because all users cannot become root and all users cannot be added to the sudoers group for security reasons, here comes the concept of setuid and setgid implementation. Let us look at the /etc/shadow file permissions.

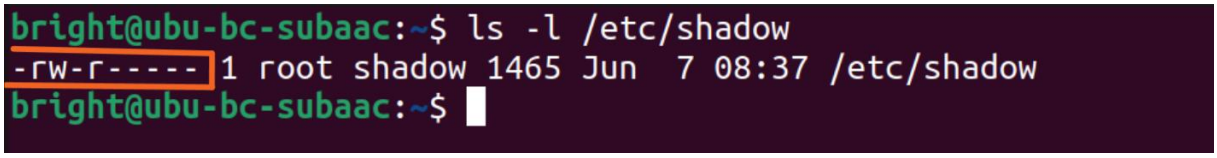


Figure 8: The file permissions of the /etc/shadow file showing write permission for root only

If you look closely at the orange-colored circled area, this is the file permission area separated with -, the first set of permissions are only read and write permissions for the file owner, which indicates here that it is a root file. The second permission, which is only read permission, is for the group shadow, which means that all members of the group shadow can only read the /etc/shadow file. The third set of permissions, which should be given to others, was restricted for security and proper administrative reasons. However, in this case, other users ought to change their passwords without having to disturb the administrators to do that every time. To achieve this, Linux has a mechanism for this: the command **passwd**, which initiates a process for normal users to communicate with /etc/shadow as privileged users. This process **passwd** has setuid permission added at owners' permission; this allows users to execute the command as privileged users and can change their passwords at their convenience.

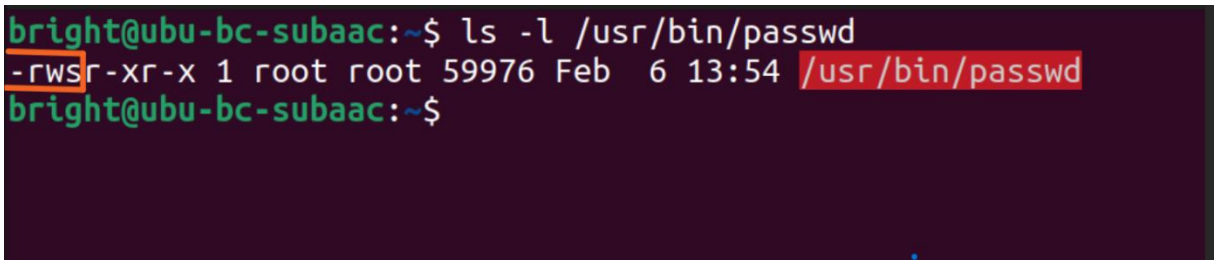


Figure 9: Setuid permission added to the command passwd, so normal users can use it to perform privileged tasks

Looking closely at the picture above, you will see the unusual **s** permission added to the permissions of the process owner, which is the root. This **s** permission allows other user accounts in the system to execute the command **passwd,** which communicates with the /etc/shadow file to change the password for the user executing the process**.** Binaries that execute using root privileges for normal users are prone to privilege escalations to obtain a shell as root when leveraged by a smart attacker. Let us demonstrate how this happens. In our Ubuntu lab

machine, we created a user named **timmy**, we started a shell for user **timmy**, then we ran the **ps** command to know processes **timmy** is executing currently as an unprivileged user, we took an instance of a process to investigate the UIDs and detect the privileges **timmy**'s processes are running on by default.

Figure 10: Normal User's Process UIDs when executing unprivileged binary

Looking closely at Figure 10 above, we will notice that the four UIDs which include the Real UID, Effective UID, Save-Set UID, and Filesystem UID, are all pointing to the user ID, which means no additional privileges were attached to the running process. Let us look at another scenario where a normal user is executing a binary file owned by root and running with root privilege. To demonstrate this, we used the **passwd** executable binary that allows normal users to communicate to /etc/shadow and change their password using root privileges. recall that 'passwd' has a setuid permission attached to it.

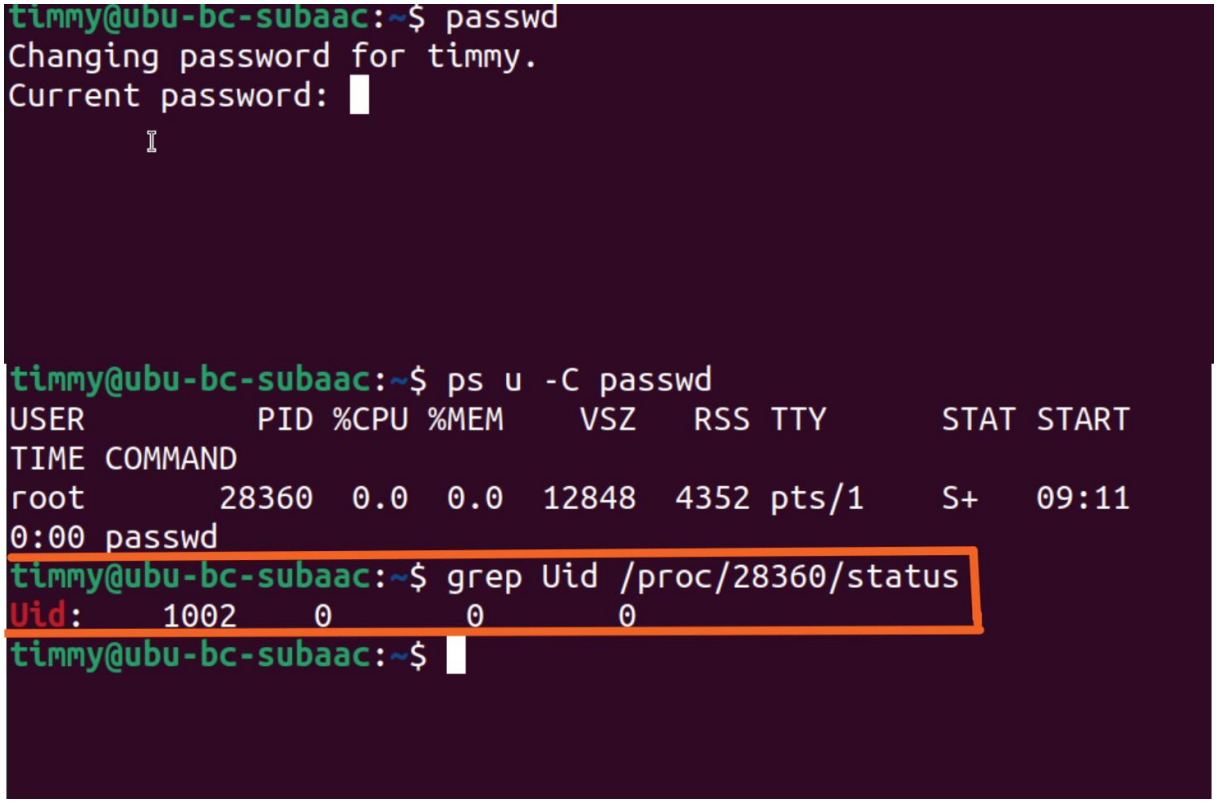


Figure 11: Normal User's Process UIDs when executing privilege binary with setuid permission

In Figure 11 above, we ran the command 'passwd' to change the password for **timmy**, recalling from our earlier explanation that the binary command 'passwd' is a root binary but can be executed by the normal user because of the setuid privilege attached to it. Looking closely at the image, we noticed something strange when we checked the second terminal where we listed the UIDs the process is running on. The Real UID is for the user, but other UIDs belong to the root, which is the command owner. This gives the best explanation for this section. This kind of scenario serves as the endpoint for cybercriminals, especially when the application with the setuid permission is not updated; some payloads can trigger its execution to obtain a root shell on the system.

Abusing Sudo: Somewhere in this document, we have mentioned sudo as a mechanism for unprivileged users to perform privileged tasks without logging in as root. Also, recall that to have this sudo privilege, a user must be added to the sudoers group in the Linux machine. Abusing sudo privilege is a common vector used to escalate privileges in a Linux environment. Linux users are always allowed all or some sudo privileges to perform some functions, and in some cases, all these privileges function as the root user would also do. If an attacker can get access to the Linux machine through a victim that is operating on sudo privileges, the attacker can use the command **sudo -l** to list the sudo privileges attached to the victim and can leverage the sudo privileges granted to the compromised user to obtain root shell in the system.

To explain this threat properly, let us leverage the example performed by Offsec (2023). An attacker got unprivileged access as user Joe on a Linux system. The attacker used the command **sudo -l** to list the sudo permissions for the user Joe.

```
joe@debian-privesc:~$ sudo -l
[sudo] password for joe:
Matching Defaults entries for joe on debian-privesc:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User joe may run the following commands on debian-privesc:
    (ALL) (ALL) /usr/bin/crontab -l, /usr/sbin/tcpdump, /usr/bin/apt-get
```

Figure 12: Using the command sudo -l to list the sudo privileges for the user Joe (Offsec, 2023)

The listing in Figure 12 shows that the user can run the crontab -l, tcpdump, and apt-get commands as a privileged user. Leveraging the **GTFOBins** (a list of Unix binaries that can be used to bypass misconfigured applications in Unix systems), offsec searched for the payload to exploit the three privileged commands and finally, obtained a root shell by abusing the apt-get application using the payload **sudo apt-get changelog apt** gotten from **GTFOBins**.

```
joe@debian-privesc:~$ sudo apt-get changelog apt
...
Fetched 459 kB in 0s (39.7 MB/s)
# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 13: Obtained a root shell by abusing the apt-get privilege binary (Offsec, 2023)

Before we conclude this section, recall that we earlier stated that Linux can be used as a domain server to host company applications such as web application servers and database servers. In some cases, most companies want to have all these servers integrated into their Active Directory environment for centralized monitoring and administration. These servers have front ends that are internet-facing to serve the needs of the public, therefore, they are prone to attack via various attack vectors. A successful attack on an integrated Linux server can grant the AD Domain Controller access. Threats application servers face today are captured in OWASP's TOP 10 Web Application Vulnerabilities (OWASP Top 10, 2021).

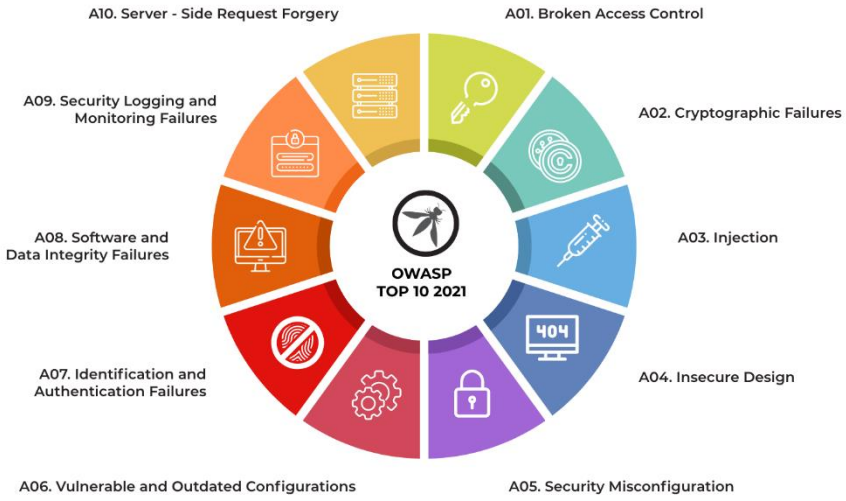


Figure 14: OWASP Top 10 Web Application Vulnerabilities (Isaiah Chua, 2022)

Let us briefly analyze these threats as exposed by OWASP's Top 10 Web Application Vulnerabilities (OWASP Top 10, 2021). We will analyze them based on the context of this document. Of course, an attacker can leverage any of these threat vectors to get a root shell in the Linux server and from there make a way into the Active Directory environment.

Broken Access Control: This happens when access to a server is misconfigured and can no longer be controlled. This type of scenario can pose a serious threat to the server. Cybercriminals can leverage this means to create root accounts and have root access to the server.

Cryptographic Failures: When data are stored in clear text or encrypted using a weak encryption algorithm. This can expose or leak sensitive data to unauthorized persons. Sensitive data can be valid account credentials that can also grant unauthorized access to the server.

Injection: Executing malicious commands as a query to access sensitive information that can grant unintended access to the server. This usually happens when the user's inputs are not sanitized or validated.

Insecure Design: When proper design architecture or threat modeling is not involved during application development, the application can behave abnormally, which might grant unintended access to the server.

Security Misconfiguration: This is all about potential vulnerabilities that were not addressed properly during application testing before rolling out, such as missing appropriate security hardenings, default account usernames and passwords, and enabling unnecessary features. Etc.

Vulnerable and Outdated Component: Developers who do not properly check their code leverage outdated components and dependencies, which can already be known or soon be known for their vulnerabilities. These outdated components can allow attackers to gain root access to the server.

Identification and Authentication Failures: When authentication or session management is weakly secured, an attacker can leverage either to gain unauthorized access to data relating to authorized users and either use them to access the server, create their account, or install a web shell or a reverse shell as a backdoor on the server.

Software and Data Integrity Failure: There will always be a time when an existing application needs an update or a patch to address existing vulnerabilities. If these software patches and updates are not properly verified for data integrity, they can create backdoors for attackers once installed into the system.

Security Logging and Monitoring Failures: Proper security logging and monitoring can facilitate the detection of malicious activities at the earliest stage, and the incident response (IR) team will respond to the incident immediately after detection. When these facilities are not in place, the attacker can move laterally within the environment without detection.

Server-Side Request Forgery (SSRF): To provide end-users with the convenience of allowing access to data via a URL has become more common. Let us think about a server supplying data for a user without first validating the URL supplied by the user, this vulnerability can facilitate an attacker to send a request from the webserver to an unexpected destination. If the server can be manipulated in such a manner, the malicious user can access unauthorized data.

2.3.2. Cyber Threat Actors

In the previous section, we discussed cyber threats in the context of Linux and Active Directory. This section will analyze threat actors, who they are, what they want, and how they execute their functions to achieve their goals. This section will also help us develop a hacker mindset, which we will implement to achieve better results during integration.

Threat actors, also known as malicious actors or attackers, represent a person, group, or organization responsible for malicious activities. They are often motivated by financial gain, political gain, or simply a desire to cause harm (Gareth Marchant, 2023).

Types of Threat Actors (Gareth Marchant, 2023)

- **Nation-State:** This type of threat actor is supported by the resources of its host country's military and security services. Its primary goals are spying and competitive advantage. This kind of attack mostly happens between countries and is generated by war or some interest.

- **Organized Crime:** This type of threat actor uses hacking and computer fraud for commercial gain. Its general purpose is to make a profit by gaining unauthorized access to an organization's network, taking illegal control of the environment, disrupting normal operations, stealing information, and holding organizations for ransom through blackmail threats.

- **Hacktivist:** This type of threat actor is motivated by a social issue or political cause. This type of threat actor uses cyber weapons to promote political agenda, releasing confidential information to the public, bringing down websites and causing denial of service attacks all to engineer political and social gains.

- **Insider Threat:** This is a type of threat actor that is assigned privileges on the system and causes either intentional or unintentional incidents. This can be an employee, freelancer, or contractor with access to the company's network. This type of threat actor can either be intentional or unintentional. Intentional insiders know and execute their actions to achieve a particular goal or objective. Unintentional Insider does so through neglect or being exploited by an external hacker.

- **Script kiddie:** This is an inexperienced, unskilled attacker that uses tools or scripts created by others. This type of threat actor performs attacks not for financial gain or any other reasonable goal but to gain attention or build technical capabilities.

HOW CYBER THREAT ACTORS ACHIEVE THEIR GOALS

To better describe this section and to help readers understand the stages of executing an attack and the tactics, techniques, and procedures used in executing an attack, we will introduce two important frameworks: **CYBER KILL CHAIN** and **MITRE ATT&CK.**

CYBER KILL CHAIN: The Lockheed Cyber Kill Chain is a framework that describes adversaries' daily work. Generally, there are seven steps that describe the order in which adversaries must complete before achieving a successful attack. An attacker will pass from one step to the other to gain unauthorized access to a victim's environment. This is a good approach to describe how an attacker can leverage the threats we discussed earlier and the threats we will discuss later to access a company's network.

Let us list the steps in order and explain the expectations of each step in the attack process:

Reconnaissance: This is the stage where the target is identified. This phase tends to gather every bit of information about the target. This is achieved by implementing information-

gathering techniques that involve both manual and automated approaches. Activities at this stage include social engineering, visiting the company's website pages, and performing scans and enumerations. Harvesting emails, looking for passwords, looking out for outdated services and vulnerable entities in the network, and visiting the organization's social media pages. A smart attacker can use these factors to gain full insight into the organization's network architecture.

Weaponization is the second stage in the attack process, where the attacker starts brainstorming on the earlier gathered information. Based on the information gathered, the attacker will start looking for the best weapon to leverage to access the environment. Depending on the attack type, the attacker will either craft payloads or purchase commercial exploits and malware. The attacker will also think of the best ways to create a backdoor in case of future connections. For phishing attacks, the attacker will select the appropriate way and appropriate file to present to the victim to convince the victim to click on the embedded links attached to the email.

Delivery: At this stage, the attacker is ready to deliver the malware to the victim(s). This can be achieved through adversary-controlled delivery directed against web servers, phishing emails, USB sticks, an employee, social media interactions, and direct attacks. Whichever way the attacker delivers the payload must be in reaction to what they tend to achieve because delivering a payload in the right format engineers a greater percentage of a successful attack.

Exploitation: At this stage, the payload has been received and triggered by a vulnerability, which might be a human being (clicking on a malicious link and attachment), application, service, operating system, port, unsecured network interface, outdated devices, etc. The attacker will use the delivered payload to exploit any of these threats to gain unauthorized access to the primary environment.

Installation: This is the stage where the adversary tends to create a long foothold in the environment, either by installing a web shell for web servers or reverse shell or binding shell malware for other environments to have backdoors for persistent actions. Most adversaries manipulate their malware to appear as if it is part of the victim's environment installation so that it is not easily detected.

Command & Control (C2 / C&C): In this stage, the attacker leverages the installed weapons in the victim's compromised environment to open a command-and-control channel. At this point, the attacker has already gained privileged access inside the victim's environment and can remotely perform commands.

Actions on Objectives: This is the final stage of the attack, where the attacker executes their objectives, such as burning down the victim, exfiltrating data for ransomware, disrupting business processes, escalating privileges, modifying and corrupting data, etc.

The image below from Lockheed Martin summarizes these descriptions in an ordered way and shows the seven stages of the attack process in a chain form.

Figure 15: The Cyber Kill Chain (Lockheed Martin)

MITRE ATT&CK: This is another important high-level framework used to gain an understanding of attackers. In a more general way, it explains the Tactics, Techniques, and Procedures in which attackers perform their attacks.

Currently, MITRE ATT&CK has different attack profile matrices, which are three in number. The three matrices are ATT&CK for Enterprise, ATT&CK for Mobile, and ATT&CK for ICS (Industrial Control Systems). The focus of this document relates to the ATT&CK for Enterprise matrices because, in this document, we are not discussing nor simulating mobile device environments, energy manufacturing, critical infrastructure environments, and other IoT devices. The ATT&CK for Enterprise matrix also has various sub-matrices. These are the PRE, Windows, macOS, Linux, Cloud, Network, and Container matrices. For this master thesis, we will center our idea on adversary emulations on Linux and Windows (Active Directory) because they are the two technologies and platforms we are discussing in this document (MITRE ATT&CK. 2024).

Adversary Emulation on Windows (Active Directory) and Linux Platforms Using MITRE ATT&CK: We will discuss both together because both are traditional operating systems; though built on different platforms, their tactics are the same, but there are slight differences in techniques (MITRE ATT&CK Widows Matrix, 2024. & MITRE ATT&CK Linux Matrix, 2024.). Active Directory is built on the Windows platform; therefore, we will describe it using Adversary Emulation on Windows. As described earlier, Linux was built on the Unix platform, and because there is already a direct MITRE ATT&CK adversary emulation for Linux OS, we will leverage it for further discussions.

According to (MITRE ATT&CK Widows Matrix, 2024. & MITRE ATT&CK Linux Matrix, 2024.), There are 12 procedures adversaries follow to achieve their evil aims on these technologies, including:

Initial Access (ID: TA0001): Initial Access includes techniques that use various entry attack surfaces to gain their entrance within a network. Some of the techniques used to achieve entrance or foothold include targeted spear phishing, exploiting misconfigurations and weak passwords, and exploiting weaknesses on public-facing web servers. Based on the adversary's aim, gaining an initial foothold through initial access may allow for continued access.

Execution (ID: TA0002): Execution includes techniques that achieve the adversary's malicious payload running on the local or remote target. The techniques associated with these tactics allow adversaries to achieve their goals through malicious code execution. For example, when an adversary executes a malicious script that uses HTTP traffic to exfiltrate data from a remote or local victim.

Persistence (ID: TA0003): Persistence includes techniques that adversaries use to retain access to compromised systems so that effects from system restarts, credential changes, and other interruptions would not cut off their access. These persistence techniques include accesses, actions, installed shells, and configuration changes that let adversaries maintain their foothold on compromised systems, such as replacing legitimate services, adding startup scripts, or installing web shells.

Privilege Escalation (ID: TA0004): Privilege Escalation includes techniques used by adversaries to gain higher-level permissions on a compromised system. When an adversary gains an initial foothold on a compromised victim via unprivileged access or through an unprivileged account, it will require elevated permissions such as root, admin, or some privileged account access to perform actions and objectives. They achieve this by taking advantage of other vulnerabilities and misconfigurations.

Defense Evasion (ID: TA0005): Defense Evasion includes techniques adversaries use to block detection as they perform their objectives in the compromised environment. Some of the techniques adversaries use for defense evasion include but are not limited to uninstalling/disabling security features or obfuscating/encrypting payloads so signatures embedded in the security software will not detect the malicious payload. Adversaries also leverage trusted services to masquerade their malware.

Credential Access (ID: TA0006): Credential Access includes techniques adversaries use to steal credentials like account names and passwords. Some of these techniques include keylogging or credential dumping, but they are not limited to them. When adversaries have access to legitimate credentials, they can access the target system as authorized users, making them harder to detect and providing more opportunities for them to create more accounts to help achieve their goals.

Discovery (ID: TA0007): Discovery includes techniques adversaries use to understand the compromised system and related networks better. This will help adversaries orient themselves before performing actions on the victim. The technique adversaries always apply here is often leveraging the native operating system knowledge, which can facilitate gathering information about a compromised operating system.

Lateral Movement (ID: TA0008): Lateral Movement includes techniques that adversaries use to move from one technology to another within a network. Sometimes, when performing attacks, the system that granted the initial foothold on the network might not be the target system; based on this, more techniques need to be applied by adversaries to move laterally within the environment until they reach their primary target. Adversaries might install their remote access tools to accomplish Lateral Movement or use dumped legitimate credentials with other stealthier networks and operating system tools to achieve this.

Collection (ID: TA0009): Collection includes techniques adversaries use to discover the type of data and information to steal from the target following their objectives. The main targets are always information on the disk drive, browsers, audio, video, and email. Some collection methods include capturing screenshots, keyboard input, etc.

Exfiltration (ID: TA0010): Exfiltration includes techniques adversaries use to steal data from compromised systems. These techniques are packaged to avoid detection. In most cases, adversaries might compress or encrypt the data before exfiltrating it. Some techniques used include HTTP traffic, command and control channels, etc.

Command and Control (ID: TA0011): When analyzing the cyber kill chain, we discussed this technique. The MITRE ATT&CK explained it as consisting of techniques adversaries use to communicate with the compromised systems within a victim network. This could be achieved by mimicking normal and expected traffic to avoid detection. Many ways are available that adversaries can leverage to achieve this more stealthily, though it will depend on the victim's network structure and defenses.

Impact (ID: TA0040): Impact includes techniques that adversaries use to stop availability or disrupt the integrity of a compromised system by making it unavailable for its original purpose in the enterprise. Some of these techniques include, but are not limited to, data tampering and destruction, ransomware attacks, etc. In some cases, though the business operations may look fine, in the real sense, they may have been altered to achieve the adversaries' goals.

We have performed an in-depth literature review on the threats associated with the technologies we are discussing in this document. We have successfully gathered the major cyber threats Active Directory and Linux face. We have also investigated how attackers can leverage these threats to obtain root shells from these technologies. Understand that we have been viewing Linux hosts from two perspectives: as a traditional Operating System and an application-hosted server; this is why we did not limit our descriptions to the threats associated with Linux as a traditional operating system, but we also leverage the **OWASP Top 10 Web Application Vulnerabilities** framework to perform threat intelligence on our Linux host. Finally, we used the **CYBER KILL CHAIN** and **MITRE ATT&CK** as adversary emulation frameworks to understand the attackers' brains.

Next, let us investigate the current administrative and security measures implemented on these technologies. As we did earlier when discussing the threats, we will also follow the same procedure by discussing the security measures differently.

2.3.3. System Administration and Security on Active Directory

This entails the overall management and configuration that administrators perform in an Active Directory environment to ensure efficient operations and security within the enterprise network. In this section, we are interested in discussing administrative measures taken in an Active Directory environment concerning security. These administrative measures are in place to serve as a protective mechanism for the Active Directory environment. We will discuss them in detail in this section.

More research and questionnaires indicate that in recent years, most companies have implemented the following administrative measures to protect and secure their Active Directory environment:

Employee Training and Awareness: This is one measure companies leverage to equip their employees to understand their responsibilities regarding cyber security. This training helps companies ensure that their employees understand security risks that can arise from

employees' actions that might threaten the company's security posture. Most companies do this in most cases annually, and the topics most regularly discussed during this training are phishing, social engineering, password protection, and physical security. One type of security threat this training can help to reduce is unintentional insider threat.

Principle of the Least Privileges: This is a security measure most companies implement to cause users and entities only to have access to the application and resources available for a particular task. Though we will explain this concept better in subsequent sections, for now, it is important to understand that this least privilege principle is achieved when resources are accessed with the correct permissions, which entails permissions given to users based on the following: **Mandatory Access Control (MAC):** Limiting access to resources based on their level of sensitivity. Such access is only granted to authorized users who need those resources to perform their jobs. An example of this access is a company's financial records, which are only accessible to the company's financial manager and auditor. **Discretionary Access Control (DAC):** Administrators grant data access to a specific number of users or groups that own that data. An example of this access is a user using another user's access right or group right to access data belonging to that user or group. **Role-Based Access Control (RBAC):** Granting access to the enterprise network based on individual roles. An example of this access is giving a writer and a reader permission to read and write files. Generally, in an Active Directory environment, the privileged users are the domain administrators, and they have complete rights over the domain controller(s), forests, and objects within the Active Directory environment. Other users are granted permissions based on MAC, DAC, and RBAC.

Disabling Legacy Protocols: These legacy protocols are default Windows server security protocols that threaten the server environment. Examples of some of these legacy protocols are using Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) to resolve host names whenever the DNS server is down. Moreover, because some of these protocols do not exist on the server, an attacker can impersonate these non-existing resources to capture hashed credentials and relay them via SMB to have privileged access to the server. Another example of these legacy protocols is enabling the IPv6 address protocol by default. When IPv6 is being used alongside an IPv4 address, without implementing more security features or disabling the IPv6 by default, an attacker can respond to a DHCPv6 request, and the attacker's IP will become the DNS server for the victim. Current security measures on the Active Directory server support disabling these protocols and enabling only the used protocols.

Disabling Clear Text Credential Cached: More often, Windows tends to make its usage simpler, so in many cases, it uses the WDigest protocol to store users' credentials in the memory. This WDigest protocol stores users' credentials as clear text to avoid prompting authentications whenever a user wants access to a file. It does this by setting the **UseLogonCredential** to 1 in the **HKEY_LOCAL_MACHINE** Windows registry. Storing users' credentials in a clear text format is not a recommended security measure; therefore, it is recommended that the UseLogonCredential settings be set to 0**.** This will cause the system constantly to prompt users

to authenticate before access. However, before achieving that, you must install the newly released software patch, the KB2871997 patch. After installing this patch, you can implement the changes. Also, note that this change is automatically present in the newer version of Windows.

Patching the Active Directory Group Policy Preferences: Lately, Microsoft released patch KB2962486 to solve the problem of elevation of privilege when using Group Policy Preferences to distribute passwords in the Active Directory. Microsoft noticed an attacker could leverage this elevated privilege to access stored passwords in the group.xml file in the Group Policy Preferences. Further recommendations also entail that the old group.xml file ought to be deleted after implementing the upgrade. If not done that way, the passwords from the old Domain Controller can still be leveraged by the attacker to control and access objects in the upgraded environment.

Leveraging Strong Password Policy: Active Directory passwords can be described as a front gate that can grant access to various activities within the Active Directory environment. They act as the first line of defense against cybercriminals. Companies, knowing that weak passwords can cause much harm to their businesses, have recently commenced the implementation of strong password policies. Generally, most companies' Active Directory robust password policy implementation includes passwords having 12–16-character length, symbols, numbers, upper-case letters, and lower-case letters.

Implementing Security Monitoring Tools and Anti-Virus Software: These are devices installed within a company's network to monitor incoming and outgoing traffic in the company's environment. This monitoring detects and prevents anomalous activities within the company's network. They are programmed with rules and standards based on the company's security policies. In some cases, they are enriched with a database of known vulnerabilities and signatures to detect anomalous activities that are true positives. Most companies using Active Directory to manage their enterprises implement some of these security monitoring tools to monitor intrusion and secure their environment from intruders. These tools include firewalls, IPS, IDS, EDR, XDR, SIEM technologies, etc. They also implement antivirus software such as Microsoft Defender. This antivirus is installed in their Active Directory environment to detect malware and viruses that entered the system through phishing emails or downloaded software.

Implementing Backup and Recovery: This is another best practice implemented by companies using Active Directory. This entails building a replica of your working environment in another environment for business continuity in the case of destruction in the original environment. Most companies perform their Active Directory backups at 60-day intervals because subsets and attributes of deleted objects, known as tombstones, stay up to 60 days before leaving the system entirely. The ntdsutil command-line tools are always used when performing recovery.

2.3.4. System Administration and Security on Linux

As discussed in the previous section, we will also elaborate on the current administrative

measures performed on Linux hosts to increase the security posture of the organization implementing the hosts for the day-to-day running of their business. It is important to understand that some of the security measures stated in the previous section also apply to this section; therefore, we will not repeat them in this section. Our interest in this section will be mostly on current administrative and security measures for Linux hosts. Let us discuss them one after the other.

Implementing Linux PAM: PAM, which stands for Pluggable Authentication Module, is a system that allows Linux administrators to configure how to authenticate users into the system. Admins can navigate to the /etc/pam.d/ directory and choose the PAM authentication program they want to use for user authentication.

```
bright@ubu-bc-subaac:~$ ls -l /etc/pam.d/
total 124
-rw-r--r-- 1 root root  384 Nov 11  2021 chfn
-rw-r--r-- 1 root root   92 Nov 11  2021 chpasswd
-rw-r--r-- 1 root root  581 Nov 11  2021 chsh
-rw-r--r-- 1 root root 1312 Jun 13 12:30 common-account
-rw-r--r-- 1 root root 1300 Jun 13 12:30 common-auth
-rw-r--r-- 1 root root 1778 Jun 13 12:30 common-password
-rw-r--r-- 1 root root 1458 Jun 13 12:30 common-session
-rw-r--r-- 1 root root 1435 Jun 13 12:30 common-session-noninteractive
-rw-r--r-- 1 root root  606 Mär 18  2021 cron
-rw-r--r-- 1 root root   69 Feb 22  2022 cups
-rw-r--r-- 1 root root 1192 Jun  1  2023 gdm-autologin
-rw-r--r-- 1 root root 1342 Jun  1  2023 gdm-fingerprint
-rw-r--r-- 1 root root  383 Jun  1  2023 gdm-launch-environment
-rw-r--r-- 1 root root 1320 Jun  1  2023 gdm-password
lrwxrwxrwx 1 root root   31 Jun 13 12:21 gdm-smartcard -> /etc/alternatives/gdm
-smartcard
-rw-r--r-- 1 root root 1419 Jun  1  2023 gdm-smartcard-pkcs11-exclusive
-rw-r--r-- 1 root root 1384 Jun  1  2023 gdm-smartcard-sssd-exclusive
-rw-r--r-- 1 root root 1426 Jun  1  2023 gdm-smartcard-sssd-or-password
-rw-r--r-- 1 root root 4126 Mär 14  2022 login
-rw-r--r-- 1 root root   92 Nov 11  2021 newusers
-rw-r--r-- 1 root root  520 Aug 12  2020 other
-rw-r--r-- 1 root root   92 Nov 11  2021 passwd
-rw-r--r-- 1 root root  270 Feb 26  2022 polkit-1
-rw-r--r-- 1 root root  168 Feb 23  2022 ppp
-rw-r--r-- 1 root root  143 Feb 21  2022 runuser
-rw-r--r-- 1 root root  138 Feb 21  2022 runuser-l
-rw-r--r-- 1 root root 2259 Feb 21  2022 su
-rw-r--r-- 1 root root  330 Feb  8  2022 sudo
-rw-r--r-- 1 root root  315 Feb  8  2022 sudo-i
-rw-r--r-- 1 root root  137 Feb 21  2022 su-l
-rw-r--r-- 1 root root  119 Dez  5  2023 vmtoolsd
bright@ubu-bc-subaac:~$
```

Figure 16: The PAM authentication programs

Inside this directory, you can see a list of Linux authentication programs. The programs may vary based on Linux distributions. These authentication programs have default directives that define their arguments; these arguments have a syntax that can be used to identify their purpose and usage, though additional configuration settings can be included on any of them depending on the company's security policy and the authentication mechanism they tend to actualize.

SSH Key Pair: This authentication security policy supersedes password authentication in a Linux remote server and has made brute force attacks almost impossible. For this type of authentication, passwords are not required. The user needs to generate a key pair, one key for the remote server and the other for the local server authenticating into the remote server.

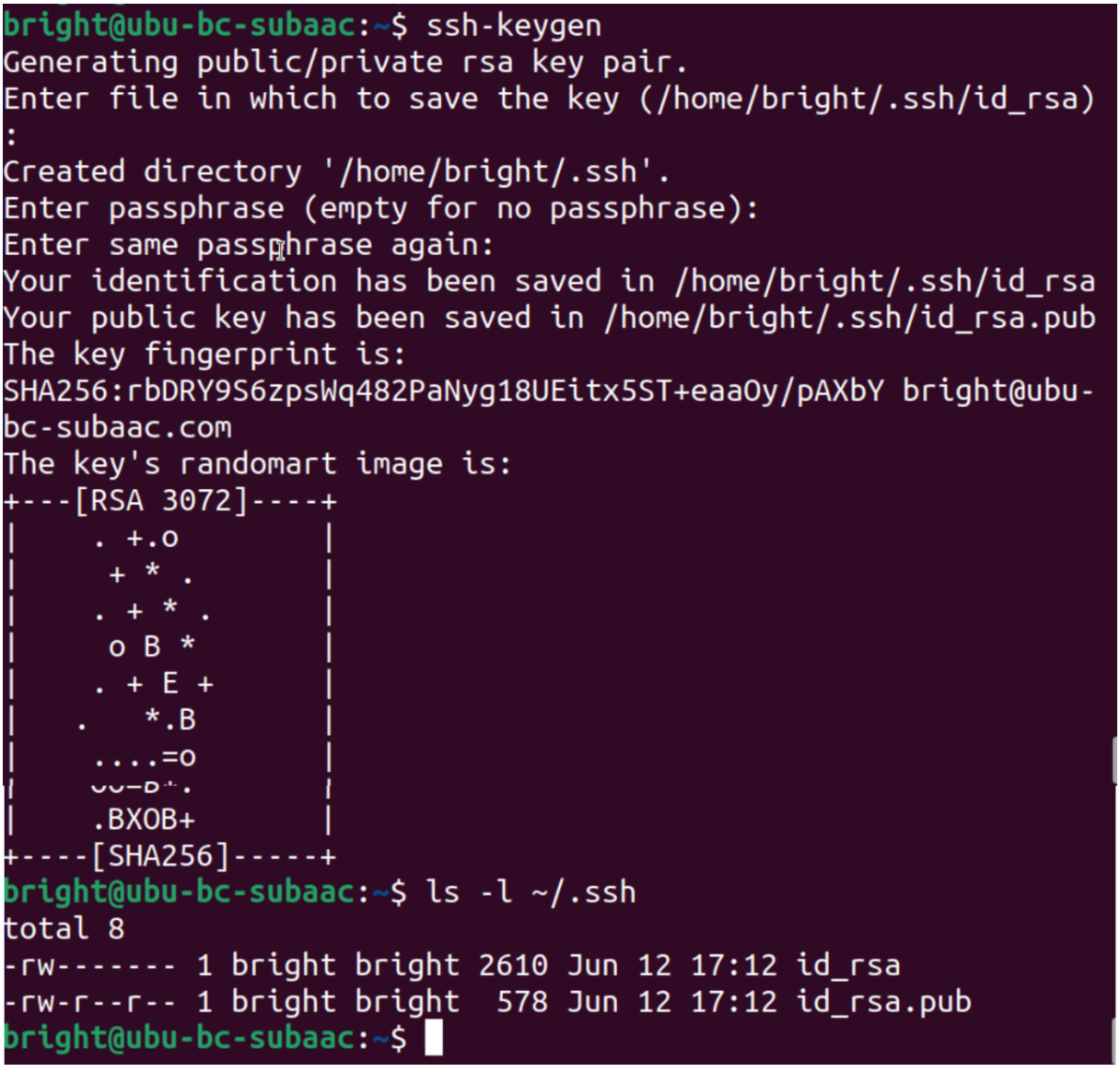


Figure 17: Generating SSH key pair

In the image above, you can see that two keys were generated: private and public keys. The **.pub key**, which is the public key, will be added to the authorized_keys file of the remote server. The remote server will use the public key to authenticate the user who has the private key, and if successful, the user will be granted access to the server without asking for a password. Most companies today using Linux have leveraged this important security feature to help their users have a secure login to their server and limit password attacks that might come through brute force, credential theft, or social engineering.

Address Space Layout Randomization (ASLR): This important Linux kernel security feature makes it difficult for attackers to predict memory addresses and redirect code execution to achieve overflow attacks successfully. When this feature is disabled by setting it to 0, an attacker can easily predict the memory address of the next instruction. However, when it is enabled and set to 1 or 2, this additional layer of security will randomize this memory address automatically, making it difficult for attackers to predict memory addresses. Most companies using Linux as their application server have implemented this security feature to protect their enterprise from overflow attacks.

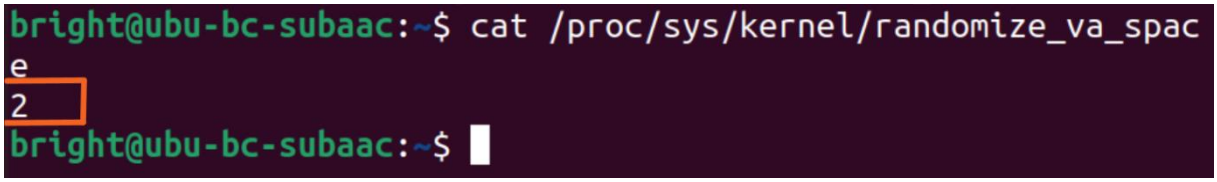


Figure 18: Showing an enabled ASLR to prevent overflow attacks

Other settings on Linux alongside ASLR for preventing overflow attacks are to set non-executable permissions to sections of the Operating System that store data and apply read-only to sections of the Operating System that execute codes.

Secure Computing (seccomp): This is a Linux security feature implemented by administrators when hosting a Linux server to limit process system calls to the kernel. Linux administrators can manipulate the type of system calls processes can make to the kernel by configuring the seccomp directory in the /proc/sys/kernel/ folder. Inside this seccomp directory, there are two files, actions_avail and actions_logged. The actions_avail file is a read-only file that returns default actions supported by the kernel, while the actions_logged file is a read-write file that returns default actions that are allowed to be logged. Companies today leverage this administrative and security mechanism to limit the types of system calls their server kernel can reply to. This makes attacks difficult for payloads that must start processes and execute some commands before taking over the system.

Let us give a practical example of achieving this security feature when hosting a Linux container. Let us move to our lab environment in our hosted Ubuntu Linux server and start a Linux container without secure computing features. We can see in Figure 19 below that we can make any system call to the kernel after hosting the container. We can try to restrict the **mkdir** system call to see what will happen.

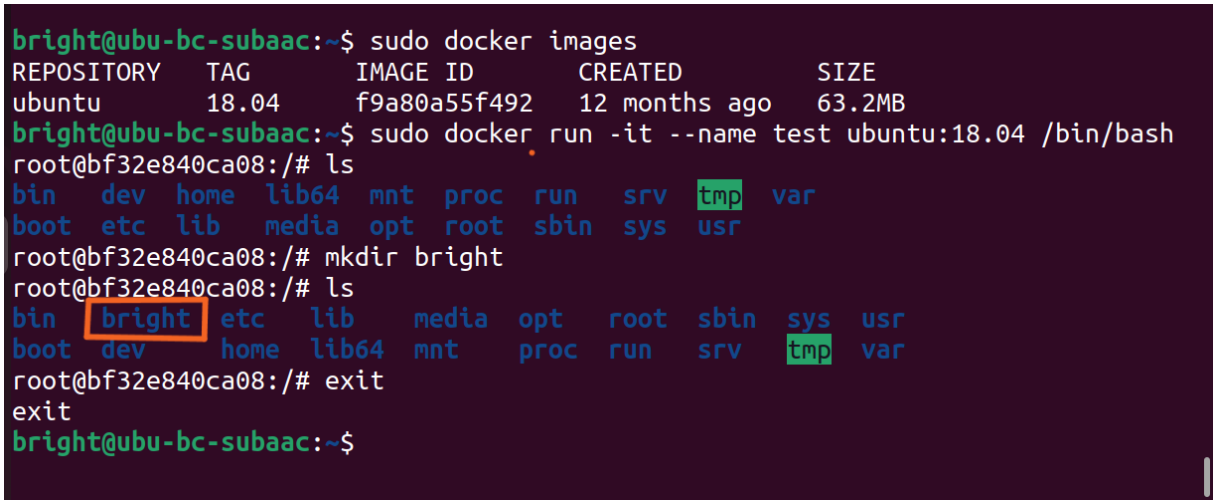


Figure 19: Linux environment without using seccomp to enforce any system call restriction

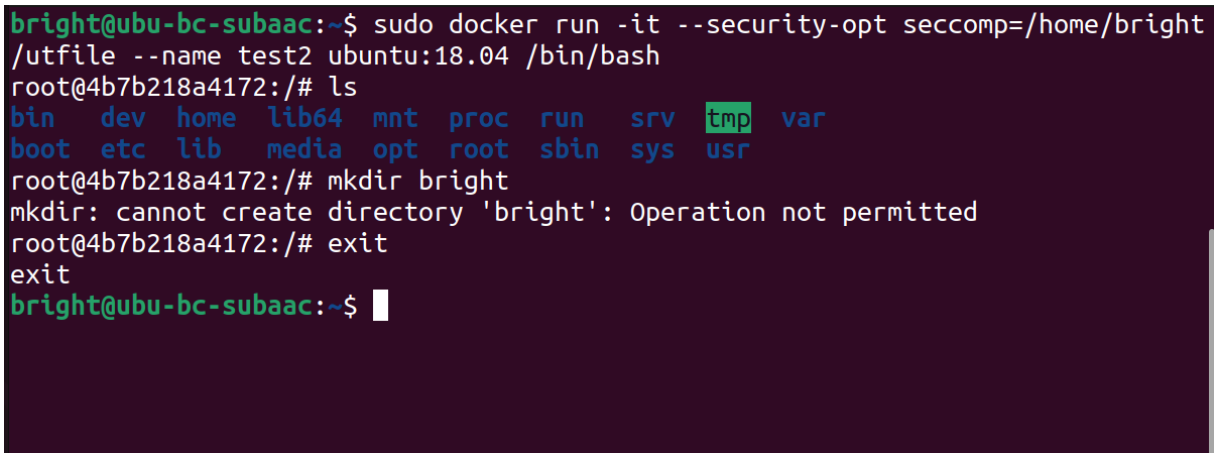


Figure 20: How secure computing (seccomp) can be applied in a Linux environment to restrict some system calls to the kernel

In Figure 20 above, the file 'utfile' in the /home/bright directory is a JSON file downloaded from GitHub (Sudip Sengupta, 2020). This JSON file contains all commands allowed by default on Linux. System and cloud administrators use this JSON file to circumvent these default settings by removing any unneeded processes or system calls and then applying it using seccomp just as we did in Figure 20; we were able to omit the **mkdir** system call.

Linux Firewalls: A firewall is a network security technology that filters traffic based on some rules configured on it. We have different types of firewall technologies, but for this work, we are only limited to Linux firewalls. Firstly, we will discuss a Linux legacy firewall technology known as Iptables. Iptables are Linux firewalls built on top of nefilters which are Linux kernel-space firewall programs. Iptables are built to evolve firewalls from the kernel space to the user space. Users can implement iptables on the Linux network to allow or block incoming and outgoing traffic in the network domain. Iptables have rules that include accept, drop, reject,

log, return, or jump to the user's specified location and can be applied to different chains that also include input, output, forward, pre-routing, and post-routing which can be found in different tables, such as filter, mango, NAT, and raw. The default table in the iptables environment is the filter table with the input, output, and forward chains, and the default policy applied to the input and output chains is the ACCEPT rule, which makes the Linux environment accept all incoming and outgoing traffic. The default policy for the forward chain is DROP. In most cases, for security reasons, these default policies are all changed to the DROP rule, while other chain rules can be applied to different chains in different tables based on the type of traffic, the source and destination of the traffic, and the purpose of the traffic. We will not elaborate more on other tables and their features, since the scope of this research work is not to discuss iptables as a topic.

```
bright@ubu-bc-subaac:~$ sudo iptables -F
bright@ubu-bc-subaac:~$ sudo iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 21: The default chain policies for the FILTER table

Above, we can see a screenshot to understand how traffic rules are applied to the Linux environment using the iptables firewall. Changing the parameter **FILTER** in the command to any other table name will also give you the default chain policies for that table. Based on this, we can define other rules we want in our environment, ranging from accepting, dropping, forwarding, redirecting, or logging traffic from a specific IP, protocol, interface, source, destination, etc.

Recent versions of Linux also have the nftables, which are upgraded versions of iptables. This follows the addition of ip6tables for ipv6, ebtables for ethernet bridge filtering, and arptables for arp filtering. Both iptables and nftables though have slight differences in their commands and underlying technologies, but they are implemented with the same ideology of allowing or blocking traffic within a Linux environment.

2.3.5. Linux Integration into Active Directory by RedHat

RedHat explained the basic integration procedure for joining the RedHat Linux enterprise into an Active Directory environment. The book was divided into three parts. The first part practically explained the use of SSSD and Realm technologies and how they function during integration. The second part emphasized joining a Linux Domain with an Active Directory domain by creating, configuring, and managing a cross-forest trust environment, which is out of the

scope of our work. The third part provides instructions on how to synchronize Active Directory and Identity Management users, how to migrate existing environments from synchronization to trust, and how to use ID Views in Active Directory environments (Delehaye et al., 2024).

Generally, RedHat tried to describe the integration process and user identity access management within the integrated technologies. RedHat limited its scope to the defensive approach achieved via identity and access management perspective but did not consider hackers' perspective by involving threat intelligence and penetration testing.

2.3.6. Integrating Linux Hosts into an Active Directory Environment - Security Measures

It is important to understand that most or all the above-mentioned administrative and security measures are used in today's industries to protect these two technologies from various attacks. In some cases, even after integration, these security measures are still implemented differently because most security frameworks and literature try to relate to these technologies from a different perspective.

Furthermore, Linux is an open-source technology, its security posture depends on the user's and enterprise needs and not vendor requirements or framework. So, because of this, there has not been any concise literature, standards, or frameworks that specifically elaborated security implementation on an integrated Linux host in an Active Directory environment. This makes users leverage the traditional security measures and implement them differently on these technologies even after integration. However, in recent times, SSSD has served as an authentication server for companies that implemented this integration. This authentication server is configured in a way that it communicates with the Kerberos authentication in the Active Directory environment and only accepts users whose accounts are valid to move around both environments. Also in most cases, administrators try to manage access between both technologies by limiting access to only specific user groups, which also involves identity and access management. These are up-to-date security measures, as seen in many block posts on the internet.

Generally, the current documentation and questionnaires we gathered emphasized identity and access management implementations to build security posture after integrating Linux hosts into Active Directory environments.

In this chapter, we have broadly reviewed the literature on these technologies. Through our literature review, we have understood the current threats Active Directory and Linux face. We also took time to investigate their current security measures. Subsequently, we tried to look at these threats and security measures from an integration point of view. However, the documentation we found only mentioned a few security measures but did not emphasize attackers' perspectives to expose the threats this integration can cause. Therefore, in the next chapter of this document, we will evaluate the integration procedure, and using the adversary emulation we learned in this chapter, we will involve the attacker's mindset to detect the threats this integration can pose to a company's security posture.

3. Evaluation

Considering the existing works of literature related to this work, which mainly featured threats and cyber defenses on Linux and Active Directory differently. Therefore, at this stage, we will consider the threat from an integration point of view to provide answers to our research questions, which say:

Why do organizations that use Linux hosts always see integrating them into their AD environment as a better option? What types of threats are these organizations exposed to because of this integration, and what suggested mitigations can be in place to mitigate these threats?

To answer these questions, we will use our laboratory environment to perform this integration process and see the threats involved, how they can be leveraged, and suggested mitigations. However, before going into the major integration process, let us investigate why enterprises integrate Linux hosts into their Active Directory environments. This will help us to answer the first research question: *Why do organizations that use both technologies for different purposes always see integrating them as a better option?*

Streamlined Operations: With Linux being among the best operating systems for application hosting, most companies have their web applications hosted on a Linux server and would like to manage or monitor activities on the server via a central environment alongside other companies' information. This helps to limit confusion when all the technologies used within the enterprise are placed in one centralized domain with one log server attached to it for easy monitoring. If this is the case, then integration becomes an excellent option because, after integration, the Linux host will appear as an object in the AD domain.

Access Control Mechanisms: You can set up your admins, users, and groups' management policies from the Active Directory Domain Controller. This policy will affect access to every other technology within the Active Directory environment, including the Linux host. Single Sign-On authentication is possible in this kind of setup, and it will limit members from having different passwords for different environments. Most of these integrations are done using Kerberos, which supports single-sign-on authentication.

Business Strategies: In a situation where Linux is employed as a file server in an organization that is also managing its business using an Active Directory, it is obvious that a file server must be used within the Active Directory domain for file sharing via FTP. In such a scenario, the Linux server should be integrated into the Active Directory domain so that file sharing will only be done within the Active Directory domain network, forming another added security layer.

Employees' Choice of Operating System: Most company employees have their preferred operating system, Windows, Linux, or Mac. After setting up these systems for employees, most companies might choose to manage authentications into these systems from their Active Directory domain. In such situations, there is no other choice than to integrate the system into their AD environment.

3.1. Integration Procedure

This section provides technical insight into the integration stages. We will be integrating Linux as an Operating System (OS), but whether as a Domain Server or traditional Operating System, integrating Linux hosts into an Active Directory environment will usually pass through the same configurations we used in this document to achieve a successful integration.

In our lab settings for the integration proper, we hosted a Linux virtual machine (Ubuntu-22.4.3 version) in our proxmox environment with the IP address 192.168.1.120, a domain (hostname) name of ubu-bc-subaac.com, and within a netmask of 255.255.255.0. We used Ubuntu Linux for this integration because, during our literature reviews and questionnaires, we noticed that Ubuntu Linux has the highest number of statistics as a Linux distribution used mainly for business processes in most enterprises. Other Linux distributions can also be integrated into the Active Directory environment following the procedure we used here. To achieve an enterprise-level demonstration, we used Ubuntu Linux.

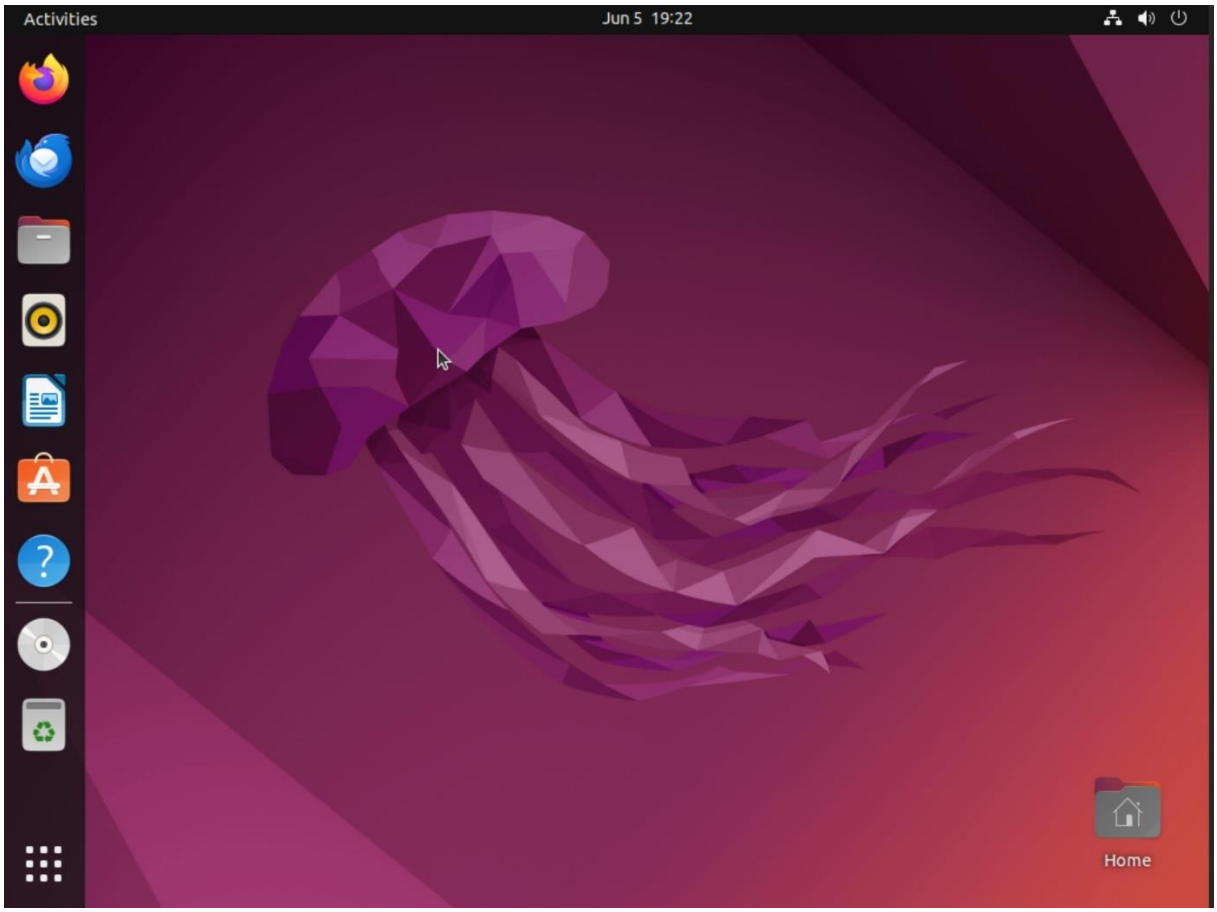


Figure 22: The Ubuntu Linux VM interface

We also hosted a Windows Server 2022 version. We configured the Windows server as an Active Directory Domain Controller with a static IP address of 192.168.1.107. The reason for

a static IP address is to ensure that the Linux host we will integrate into it will maintain the connection and not disconnect because of the change in the IP address. If the IP address is kept dynamic, it will be difficult to retain communication between both technologies, especially when the IP used by the Linux host to discover the Active Directory Domain Controller has been changed. Secondly, the AD Domain Controller IP will be used as the DNS server IP, which the Linux host will point to so that it can resolve DNS traffic. Changing this dynamically will disconnect the DNS settings between the Linux host and the Active Directory Domain Controller.

Furthermore, because the Active Directory domain requires a unique computer name, both the NetBIOS computer name and its DNS hostname (Also known as forest name) should be uniquely defined and correspond to each other. Therefore, we gave the Domain Controller a NetBIOS name of **BJ-DC1**, gave the forest a hostname **subaac.com**, and then placed it on the same netmask with the Linux virtual machine. We pointed the Active Directory's DNS server to the Domain Controller's IP address and created a login account for the local administrator.

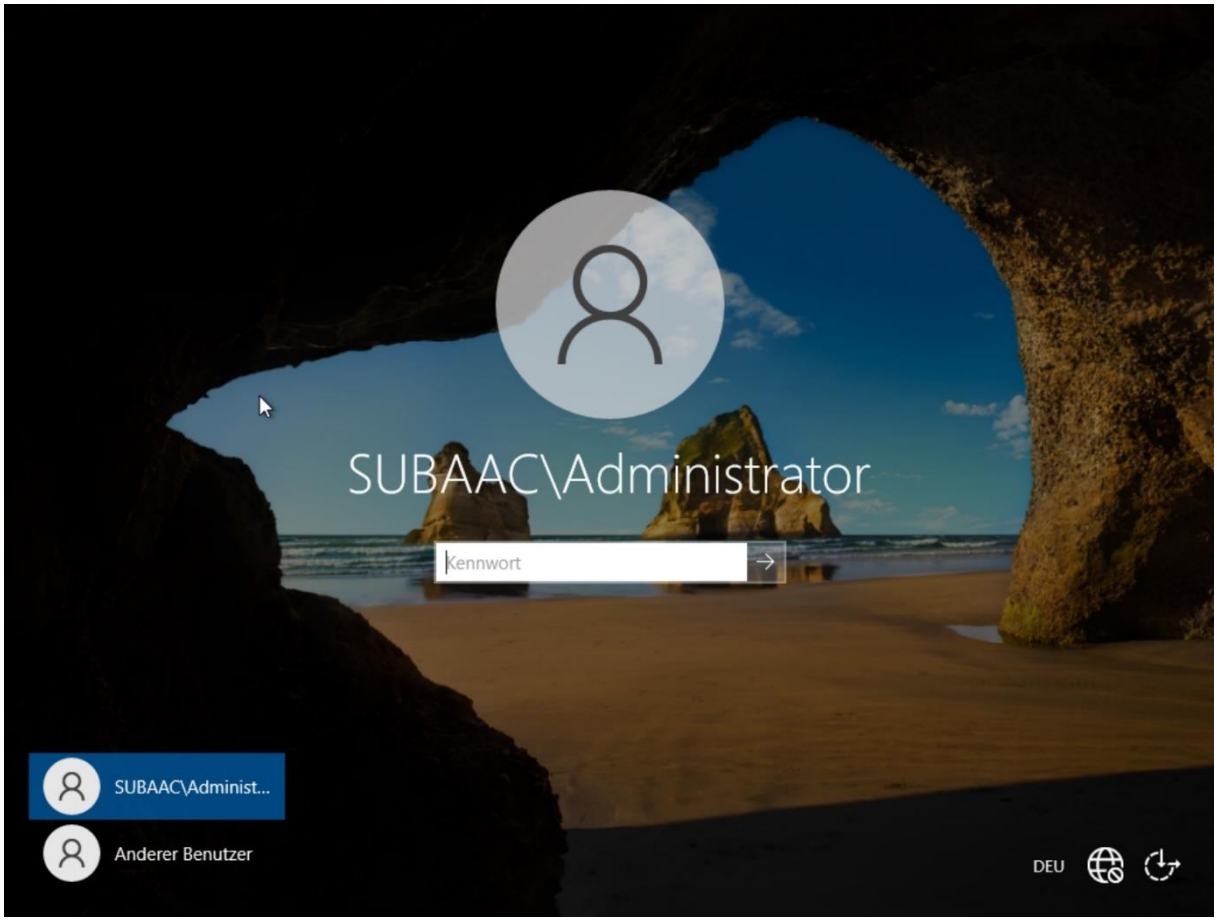


Figure 23: Active Directory Domain Controller administrative login interface

Looking at the images below, we will see the Domain controller interface and the DNS settings

after all configurations have been made. The first image is the first interface presented to a user who logs in to the domain controller, while the second image is the DNS setting of the domain controller.

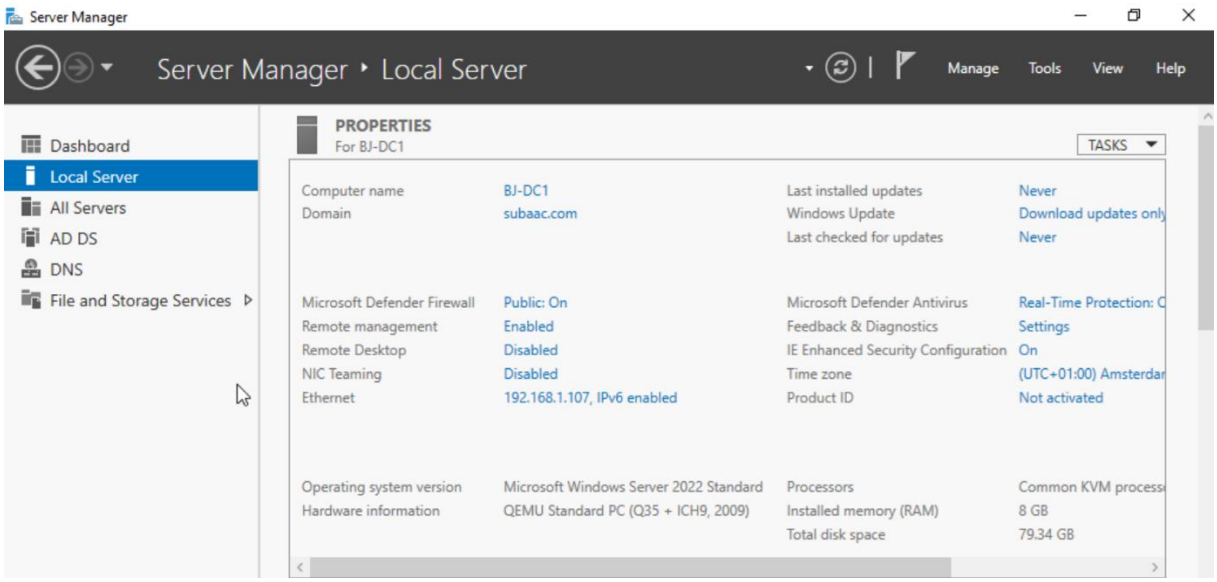


Figure 24: The Active Directory administrative domain

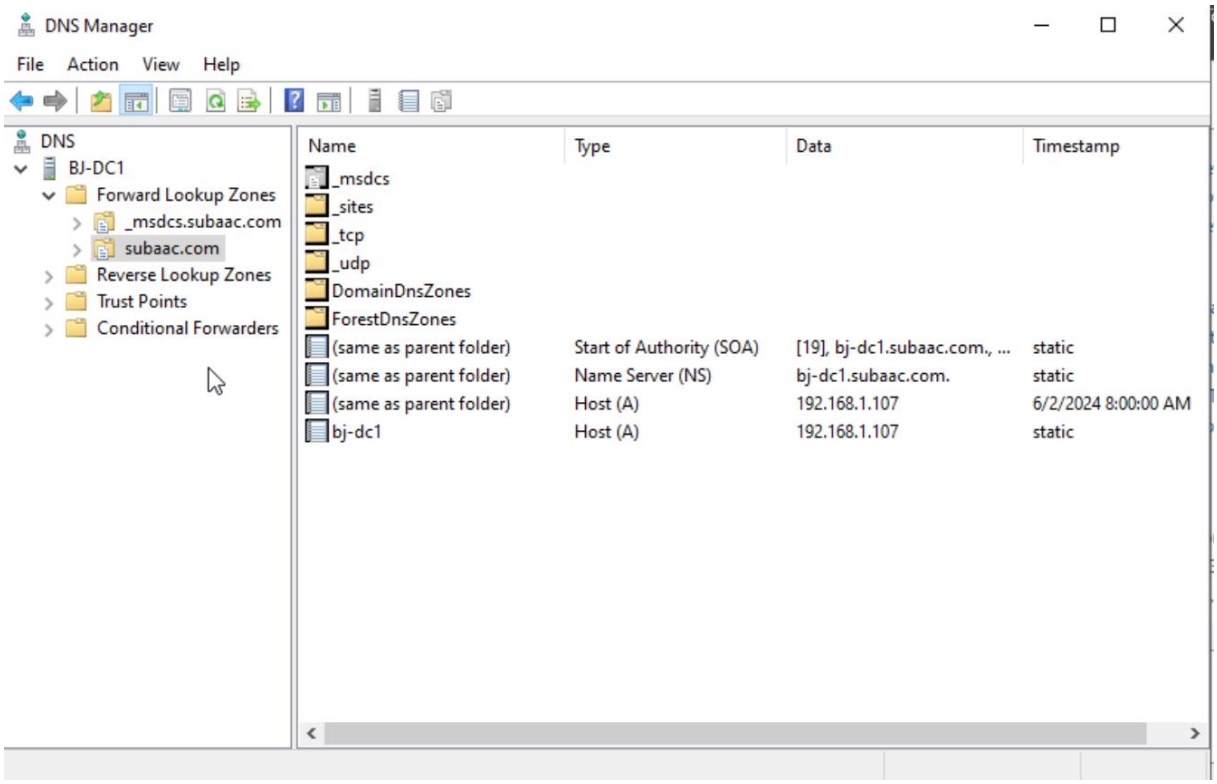


Figure 25: Domain controller DNS configuration

Before we continue, let us look at the network configuration of both technologies before integration. This integration must be possible only when both technologies are in the VLAN or subnet, pointing to the same DNS server in case you want to resolve both host names and monitor all DNS traffic from a central point. In our demonstration, we should notice that the technologies are on the same VLAN, 192.168.1.0/24. This makes it easier for them to see themselves in the network and share resources, so there will not be restrictions during the proper integration.

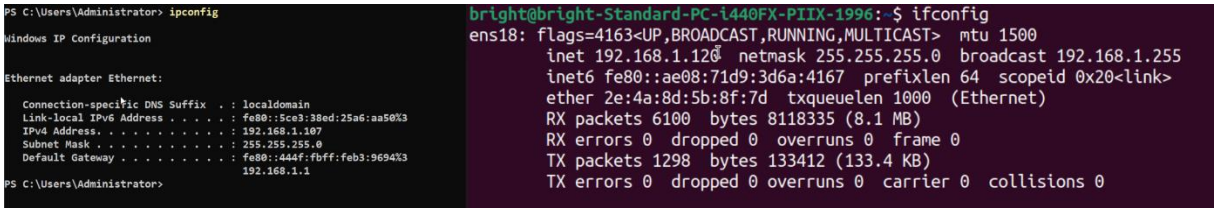


Figure 26: Active Directory and Linux network information

Since our business name is SUBAAC and we are using the domain name subaac.com, all the technologies integrated into the website should have this domain name attached to them. Therefore, we also changed the hostname of our Ubuntu Linux server to have the domain name of the Active Directory attached to it.

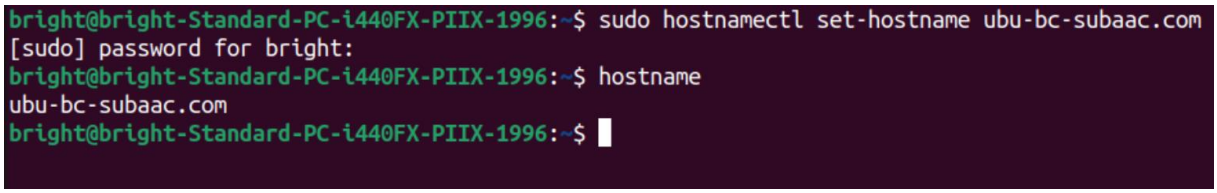


Figure 27: Set the Linux hostname to match the enterprise naming convention

Before the integration, we want to be sure that a Domain Controller is seen on our network with designated ports open on that IP address and to show that the configurations we made on the Active Directory Domain Controller are functional. Also, to know that we are currently in an unrestricted environment to perform our tasks without any form of restriction from any security device, we ran an Nmap command on that IP. In the Nmap command, we included flags to understand the host fully. We included the -sV flag to get the services running on the ports and the -Pn flag, meaning **no ping**, because we want to get our total result without performing a host discovery scan.

Understand that before integration can take place using SSSD, the following ports must be opened to allow both networks to communicate, they are DNS 53 UDP and TCP, LDAP 389 UDP and TCP, Kerberos 88 UDP and TCP, Kerberos 464 UDP and TCP Used by kadmin for setting and changing a password, LDAP Global Catalog 3268 TCP If the id_provider = ad, and Samba 445 UDP and TCP (Delehaye et al., 2024). These services running on these ports should synchronize with configurations we will perform on the Linux host to enable integration. Below is the output of our Nmap scan, which shows that the required ports are already open.

```
bright@ubu-bc-subaac:~$ nmap -sV -Pn 192.168.1.107
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-09 16:25 CEST
Nmap scan report for subaac.com (192.168.1.107)
Host is up (0.00060s latency).
Not shown: 988 filtered ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH for_Windows_8.1 (protocol 2.0)
53/tcp   open  domain?
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-06-
09 14:26:05Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
subaac.com0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain:
subaac.com0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
1 service unrecognized despite returning data. If you know the service/version
, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.c
gi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=6/9%Time=6665BB81%P=x86_64-pc-linux-gnu%r(DNSVe
```

Figure 28: Network mapping on the Active Directory host IP from the Linux host

After this discovery, we copied the domain name and the IP address of the Domain Controller into the /etc/hosts file of the Linux OS and performed a host discovery scan by sending an ICMP request packet to the host.

```
  GNU nano 6.2                          /etc/hosts
127.0.0.1       localhost
127.0.1.1       ubu-bc-subaac.com
192.168.1.107   subaac.com
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figure 29: Adding the Active Directory domain to the Linux host file

At this point, we need to do a ping scan. In the initial scan, we used Nmap and included the flag 'Pn', which means no ping. We did this to get the information from the Domain Controller without sending requests or packets. Now, we will send an ICMP echo request to it and ensure our packets are sent correctly. We should get an echo reply from the Domain Controller to be sure the host is up and running.

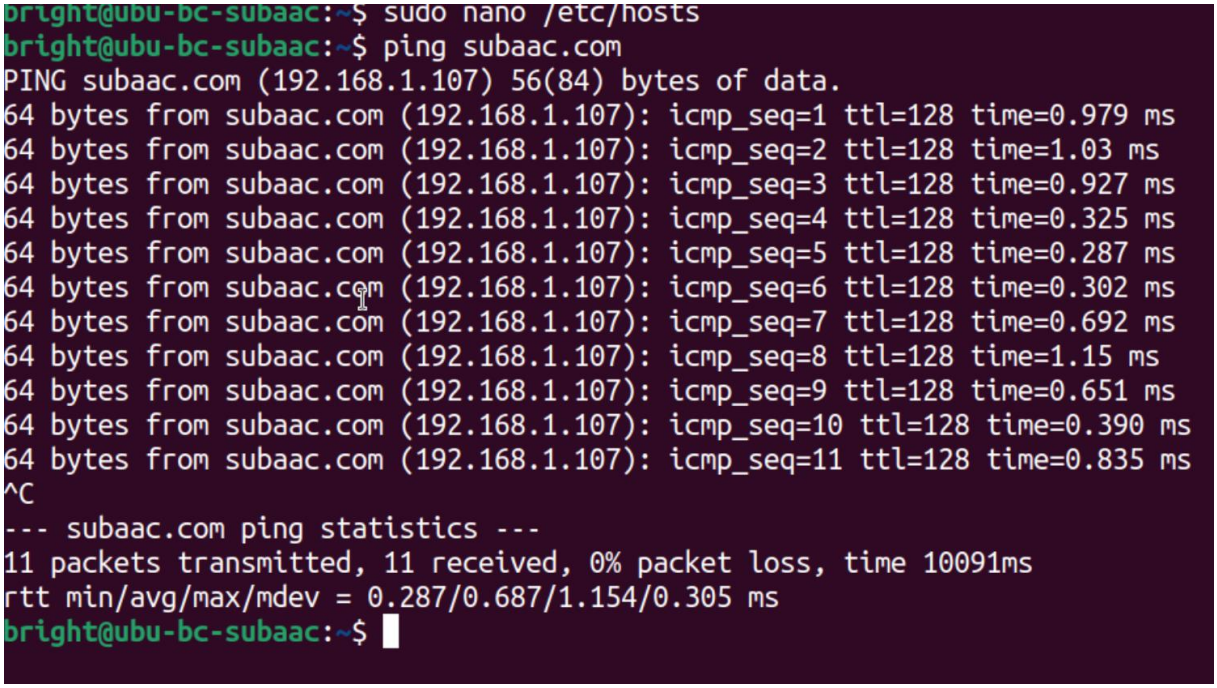


Figure 30: Pinging the Active Directory server

Looking at the image above, we should see a ping response from the Active Directory, showing that the host is alive and can receive packets, process them, and respond to the sender.

At this point, both technologies are set for integration. However, we installed different tools that helped us successfully integrate these technologies. The functions of these tools are explained below. They are currently the universal tools used to integrate Linux hosts into one domain forest, and that is exactly what we are doing in this practical. We used the primary tools we needed for the integration; other tools that function after integration are out of the scope of this work. The tools are.

Realm: This command-line utility uses Kerberos authentication to permit enrollment or restrictions in an environment such as an Active Directory domain. In this demonstration, we used realm to join the Ubuntu Linux host into the Active Directory domain.

Krb5-user: This is a Kerberos version 5 utility used to guard entrance into a domain such as an Active Directory domain. It performs password or single-sign-on authentication depending on the configuration before allowing entrance into a domain. In this demonstration, it is used during joining to authenticate the admin password we entered and be sure that it matches the

password for the admin of the AD Domain Controller. It also allows authentication via SSH during subsequent logins.

Libnss-sss and Libpam-sss: These tools provide an NSS and PAM interface, providing a set of daemons for the Linux OS to authenticate remotely into other directories and authentication mechanisms. In this demonstration, these utilities are used alongside SSSD to allow remote authentication to the AD Domain Controller using Kerberos authentication.

SSSD (System Security Service Daemon): This tool works with the NSS and PAM to enable remote authentication into either a directory service or authentication mechanisms. SSSD can allow communications to multiple domains within a single forest. We implemented it in the demonstration because our Linux host was joined to a single forest (subaac.com) in the Domain Controller. If we were joining multiple forests, we would have considered using the **winbindd** tool instead of the SSSD. Therefore, in our laboratory work, the SSSD was practically used alongside the libnss-sss and libpam-sss to allow remote authentication between the two technologies using Kerberos because, by default, SSSD can discover every domain and service within a single forest, and it also tries to resolve requests to objects in the AD domain. Another function of SSSD is providing UIDs and GIDs for AD users to log into the Linux host from the AD domain. Understand that Linux uses user IDs (UID) and group IDs (GID), while Windows uses security IDs (SID). Users authenticating into the Linux system, including AD users, must have a UID and GID assigned to them. SSSD can use the SID of an AD user with an algorithm to generate POSIX IDs in a process called ID mapping. ID mapping creates a map between SIDs in AD and IDs on Linux. Also, When SSSD detects a new AD domain, it assigns a range of available IDs to the new domain. Therefore, each AD domain has the same ID range on every SSSD client, and when an AD user logs in to an SSSD client machine for the first time, SSSD creates an entry for the user in the SSSD cache, including a UID based on the user's SID and the ID range for that domain (Delehaye et al., 2024). The package **sssd-tools** provides the tools the SSSD needs to perform its functions.

Adcli: This tool provides a command-line interface for the Active Directory within the Linux environment. This tool provides an AD interface to perform functions on the remote Active Directory domain from the Linux interface.

Samba-common-bin, Samba-common, and Samba-libs: These utilities support common file sharing between the two technologies. It is present in this integration process for future use to contain all file shares between the Linux host and the AD environment.

Oddjob and Oddjobmkhomedir: These are D-Bus (Desktop Bus) services. They serve as middleware mechanisms that allow logged-in users to communicate with existing processes within the Linux environment. The Oddjobmkhomedir utility allows the creation of a home directory for an authenticated user inside the Linux environment.

Packagekit: This service allows users to install and update software within a specific environment. In this demonstration, the package kit is a tool used to allow authenticated users to install

and update software on the remote directory service.

```
bright@ubu-bc-subaac:~$ sudo apt install -y realmd krb5-user libnss-sss libpam
-sss sssd sssd-tools adcli samba-common-bin samba-common samba-libs oddjob odd
job-mkhomedir packagekit
```

Figure 31: Installing needed tools for integration

At this point, knowing fully well that the result of this integration will cause the Linux server to synchronize its domain name with the Domain Controller DNS server, it is important at this stage to stop and disable the Linux local DNS server, then navigate to the name server section of /etc/resolv.conf, change the IP of the local DNS server to the IP address of the Active Directory Domain Controller to point the nameserver of the Linux server to the Active Directory Domain Controller. The outcome will cause the Domain Controller's DNS server to also control the DNS traffic for the Linux host.

```
bright@ubu-bc-subaac:~$ sudo systemctl disable systemd-resolved.service
Removed /etc/systemd/system/multi-user.target.wants/systemd-resolved.service.
Removed /etc/systemd/system/dbus-org.freedesktop.resolve1.service.
bright@ubu-bc-subaac:~$ sudo systemctl stop systemd-resolved.service
bright@ubu-bc-subaac:~$ sudo systemctl status systemd-resolved.service
sudo: unable to resolve host ubu-bc-subaac.com: Temporary failure in name reso
lution
○ systemd-resolved.service - Network Name Resolution
     Loaded: loaded (/lib/systemd/system/systemd-resolved.service; disabled;
     Active: inactive (dead)
       Docs: man:systemd-resolved.service(8)
             man:org.freedesktop.resolve1(5)
             https://www.freedesktop.org/wiki/Software/systemd/writing-networ
             https://www.freedesktop.org/wiki/Software/systemd/writing-resolv

Jun 05 19:10:59 bright-Standard-PC-i440FX-PIIX-1996 systemd-resolved[374]: Us
Jun 05 19:10:59 bright-Standard-PC-i440FX-PIIX-1996 systemd[1]: Started Netwo
Jun 05 19:11:04 bright-Standard-PC-i440FX-PIIX-1996 systemd-resolved[374]: en
Jun 05 19:11:04 bright-Standard-PC-i440FX-PIIX-1996 systemd-resolved[374]: en
Jun 05 19:11:04 bright-Standard-PC-i440FX-PIIX-1996 systemd-resolved[374]: en
Jun 05 19:11:31 bright-Standard-PC-i440FX-PIIX-1996 systemd-resolved[374]: Cl
Jun 05 19:25:35 ubu-bc-subaac.com systemd-resolved[374]: System hostname chan
Jun 05 20:05:34 ubu-bc-subaac.com systemd[1]: Stopping Network Name Resolutio
Jun 05 20:05:34 ubu-bc-subaac.com systemd[1]: systemd-resolved.service: Deact
Jun 05 20:05:34 ubu-bc-subaac.com systemd[1]: Stopped Network Name Resolution
lines 1-18/18 (END)
```

Figure 32: Disabling the local name server of the Linux host

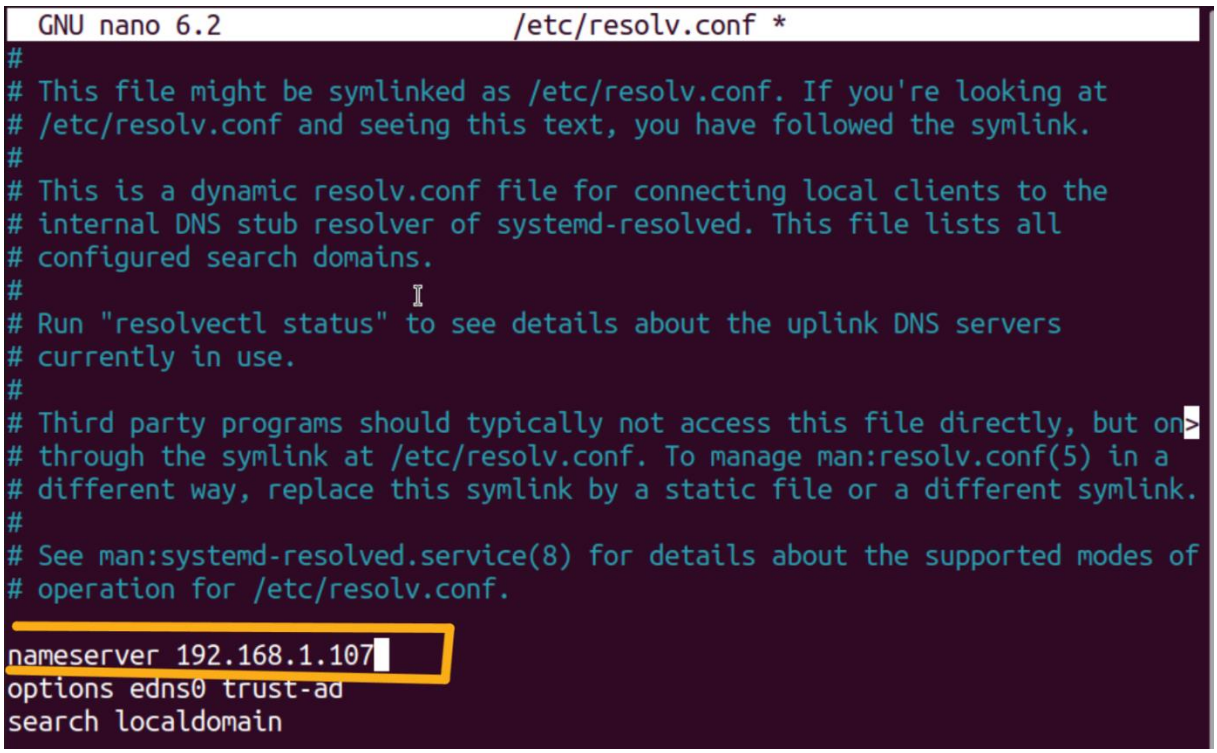


Figure 33: Pointing the Linux name server to the name server of the Active Directory

To join the Linux host to the AD domain, we first made sure that realm can discover the Domain Controller from the Linux host, so we performed host discovery using realm.

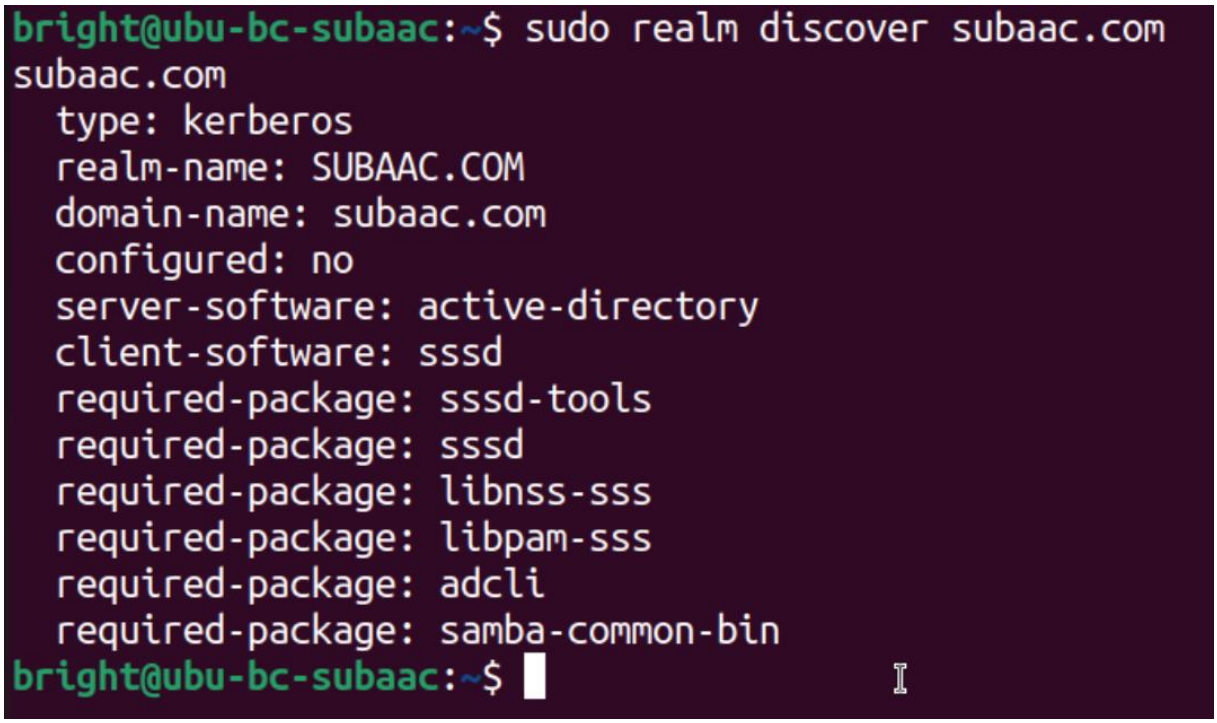


Figure 34: Using realm to discover the Domain Controller

This realm discovery process allows the Linux system to discover a domain and its capabilities. After discovering the domain using realm, we also noticed in the output of Figure 34 that the command returned a complete domain configuration and the necessary tools required to perform the joining process. Recall earlier that we discussed those tools in detail and installed them on the Ubuntu machine.

Subsequently, we used the same tool to join both technologies. We included the -v flag to get a verbose output of the internal joining process, so in case of an error in between, we can easily find it and fix it. Furthermore, we used the flag -U to indicate that we use an administrative account configured in the Domain Controller to perform the joining process.

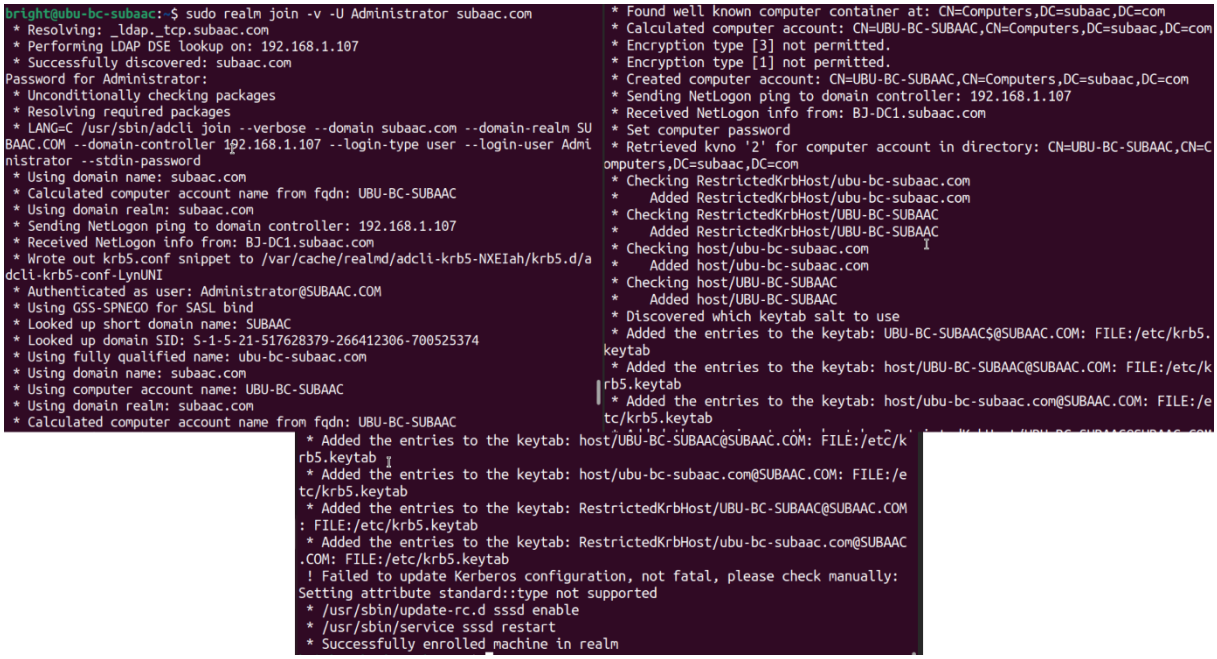


Figure 35: Verbose output of a successful Linux and Active Directory integration

The image above shows the complete integration's outcome in detail because of the -v flag added to the realm command. It discovered the domain using LDAP DSE lookup and then asked for the domain administrative password. The realm technology automated the rest of the integration process by automatically involving other packages, as specified in the output of the realm discover command in Figure 35. This helped us achieve complete and successful integration.

After successful integration, we returned to our Active Directory environment to confirm that we could find the integrated Linux host as an object. We achieved that by navigating to **Tools → Active Directory Users and Computers** and finding the name of the device we just integrated, which showed that it had been successfully added.

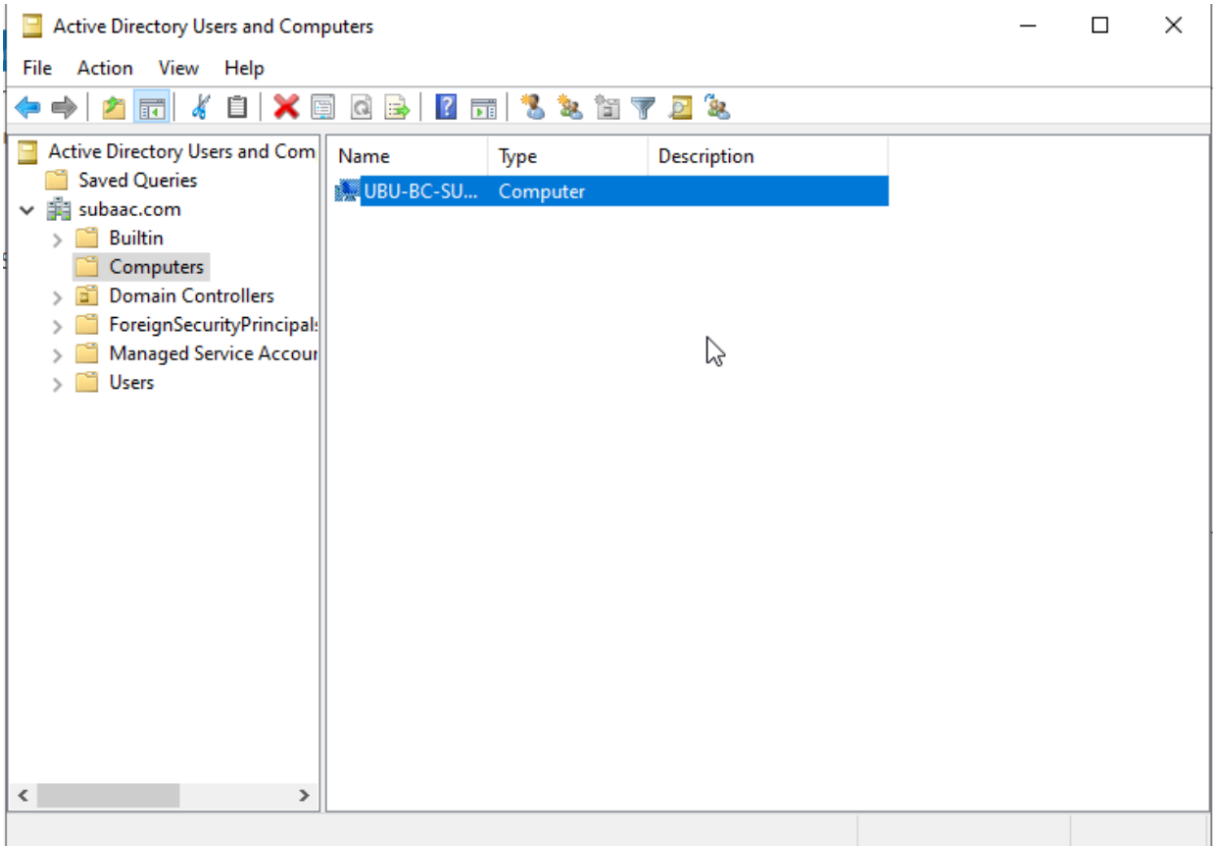


Figure 36: The Linux host becomes an object in the Active Directory environment

We can also confirm that we can log in as an administrative user and perform administrative functions via SSH connections.

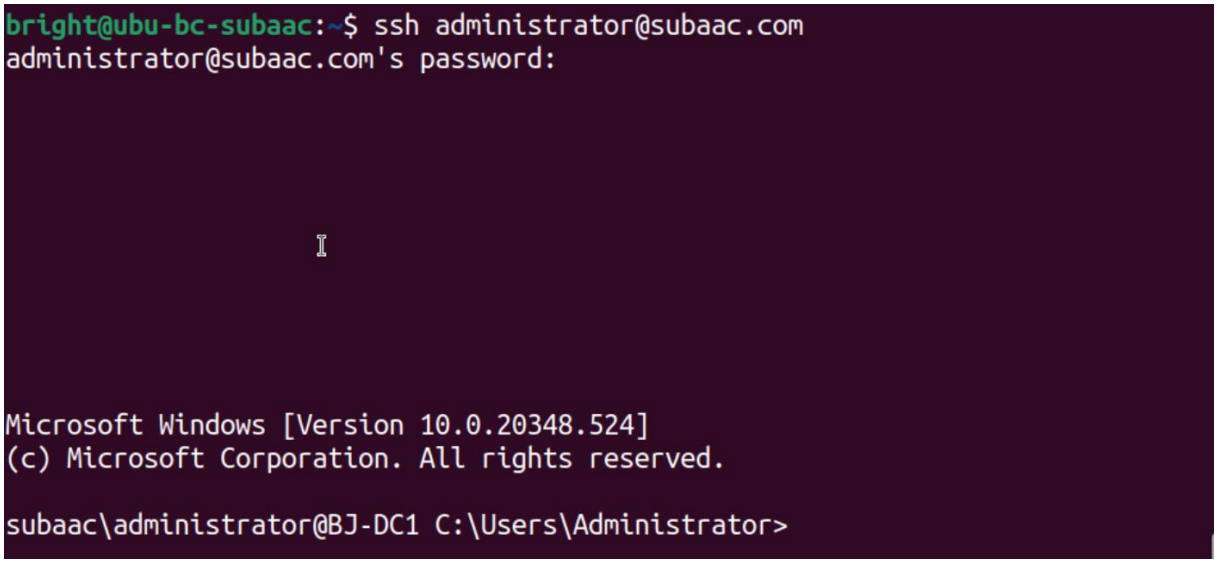


Figure 37: Getting the administrative shell of the Domain Controller from the Linux host

3.2. Testing the Authentication and Granting Sudo Access

The login we made via SSH in Figure 37 is not the general idea for this practical because our Nmap result in Figure 28 shows that port 22 is open; this means that systems within the network can have SSH access to the AD domain via port 22 (SSH).

Our major concern in this demonstration is to have our Linux host's general management (including authentication) from the Active Directory. Therefore, we will first investigate the configurations in the sssd.conf file. Recall from section 3.1. where we mention and describe all the tools we used for the integration, SSSD was one of the tools we described that it used to configure how our integrated Linux host communicates to the remote directory. On this note, let us have a look at the configuration file from the Ubuntu host to find out if it was automatically and properly configured during the integration process.

```
bright@ubu-bc-subaac:~$ sudo cat /etc/sssd/sssd.conf
[sudo] password for bright:

[sssd]
domains = subaac.com
config_file_version = 2
services = nss, pam

[domain/subaac.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = SUBAAC.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@%d
ad_domain = subaac.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
bright@ubu-bc-subaac:~$
```

Figure 38: SSSD configuration file

Looking at the image above, we notice that the file was configured by default during integration. It installed the *Pam* and *NSS* modules and started all the necessary services.

From the image above, we can observe the following things things:

- The fallback_homedir is */home/%u@%d*. This means a user will have a home directory of */home/user@domain*.
- The cache_credentials setting has been set to true. Meaning a user can still log in even if the Active Directory is unavailable.
- The *use_fully_qualified_names* has been set to true. As a result, users must authenticate using the user@domain method.

Furthermore, since the realm command did not set up the pam_makehomedir, we configured it manually to allow our login users to have a home directory after authentication.

```
bright@ubu-bc-subaac:~$ sudo pam-auth-update --enable mkhomedi
r
[sudo] password for bright:
bright@ubu-bc-subaac:~$
```

Figure 39: Manually configuring pam_makehomedir

Noticing that all configurations have been made, we now set to create users in our Active Directory environment and authenticate them into the Ubuntu client. In our demonstration, we created a user called **Sabine** and used the newly created user credentials to authenticate into the Linux environment.

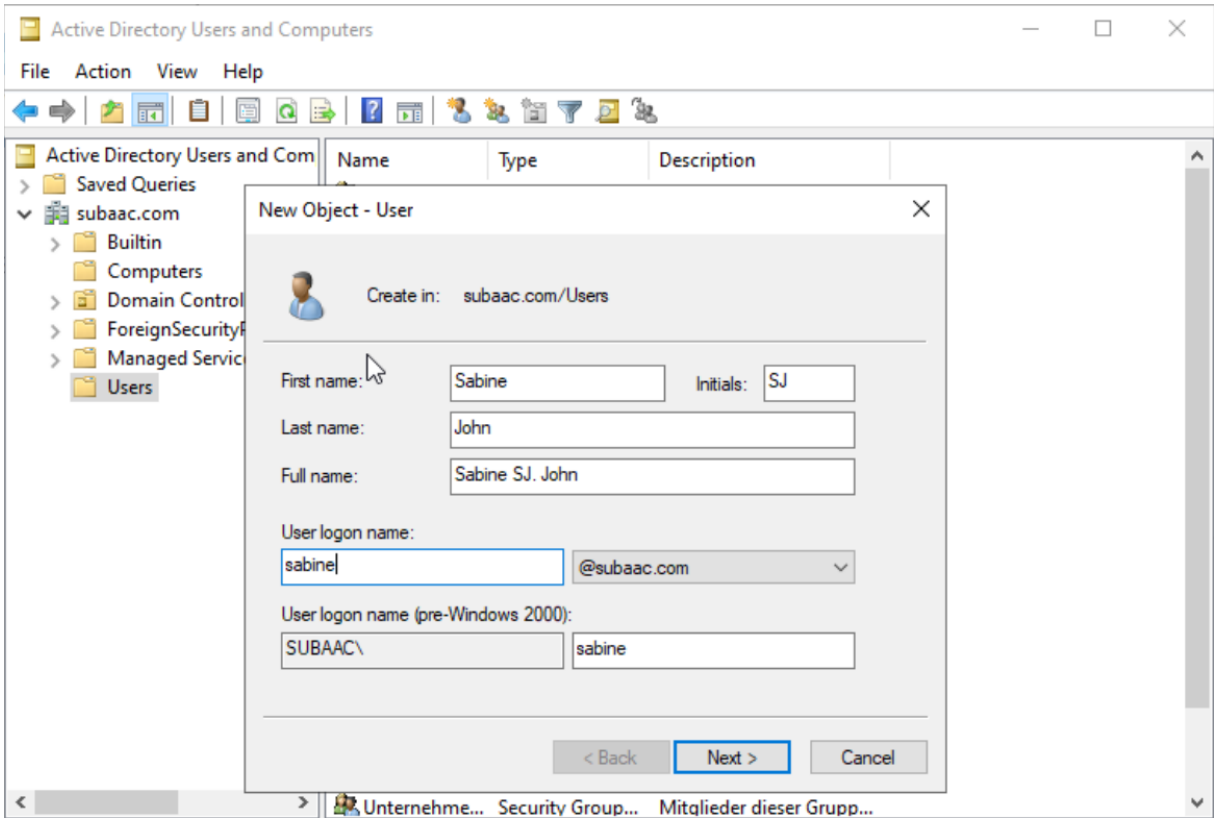


Figure 40: Creating a new user in the user's section of the Domain Controller

After creating the user **Sabine**, we tried to test the login using the newly created user's credentials via our Ubuntu command line. We entered the username in the format user@domain as specified in the sssd.conf file, and with the user's password, we successfully authenticated. The new user's home directory was also automatically created. Finally, the account was automatically added to the login window of the Linux host and was seen during login. This indicates successful integration since we can navigate both technologies using one credential.

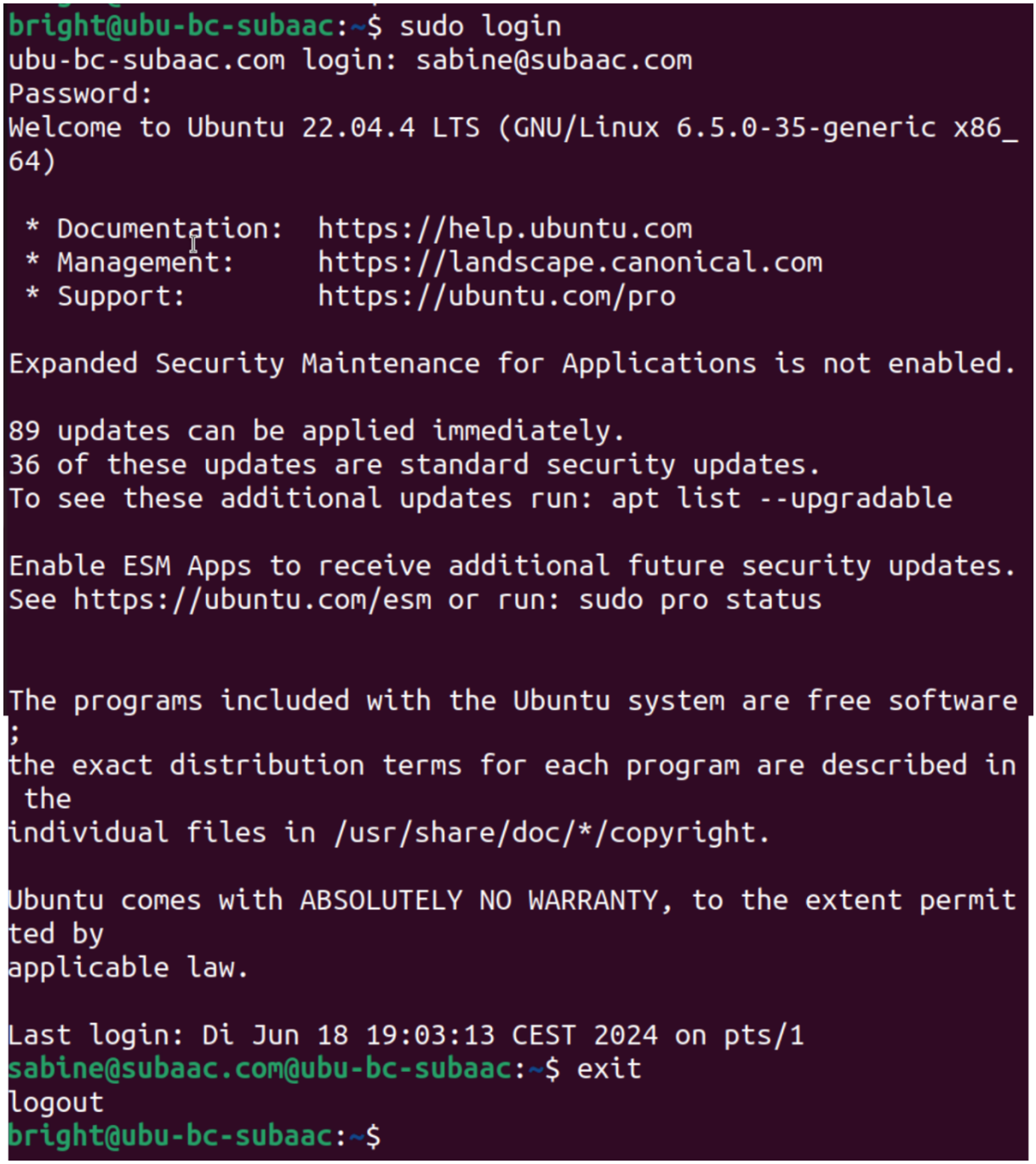


Figure 41: Testing the newly created domain user via our Linux terminal

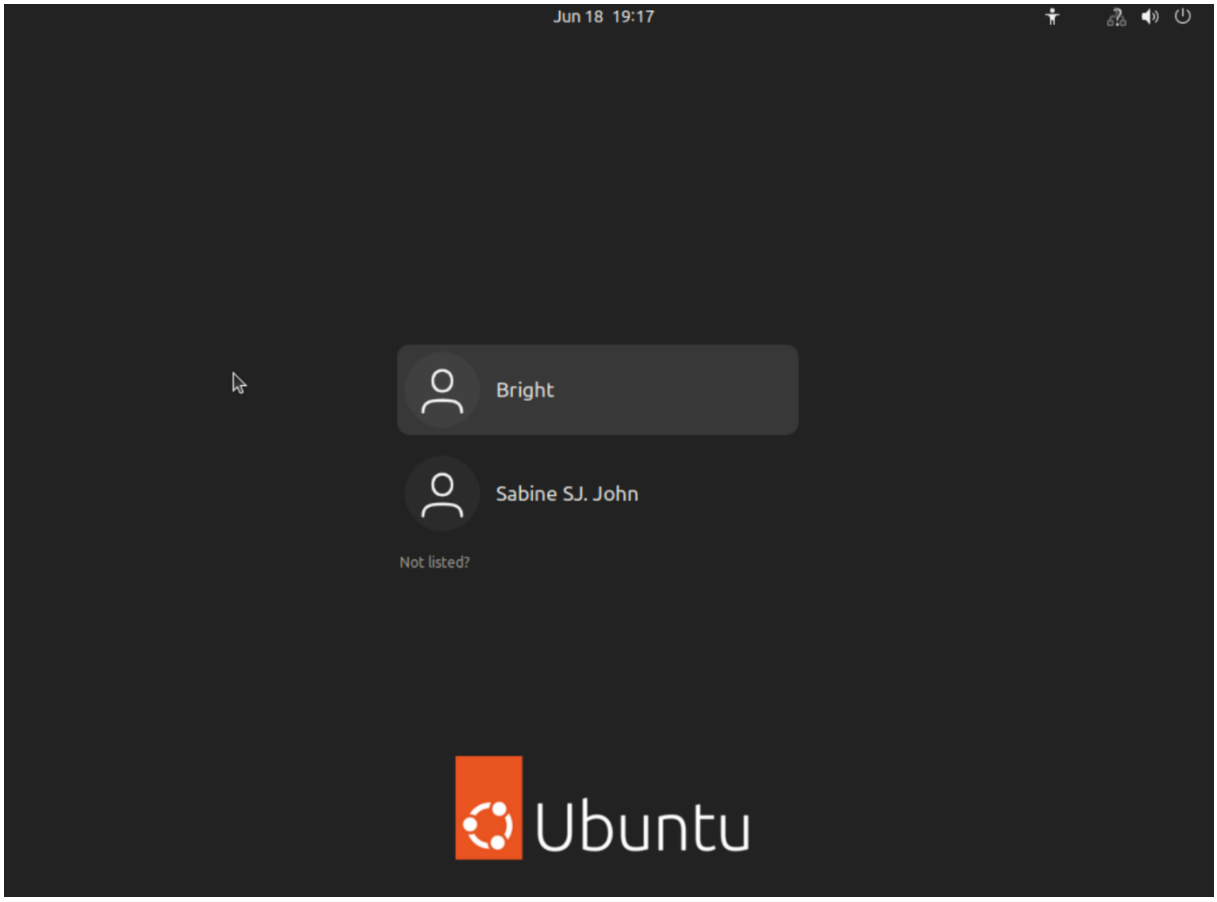


Figure 42: The user has a home directory in the Linux host, and the account is seen on the login window of the Linux host

Now that the integration is successful, the domain users we created in our Active Directory Environment can authenticate into the Linux system using their Active Directory account. After authentication, they will be provided with a home directory.

3.2.1. Granting Sudo Access

Next, we will look into granting sudo access to authenticated users; recall that for easy flow within a Linux environment, sudo access is required because most of the commands used in the Linux terminal can only run using root permission while accessing the Linux environment as root is not an action that should be used very often by every user, this is where sudo access comes into play so that normal accounts can use sudo to perform privilege actions. In this demonstration, we created a sudo configuration file named domain inside the sudoers.d directory; using an editor, we opened the file and gave sudo permission to our domain users by allowing them all permissions to perform privileged actions. The term domänen-benutzer in the image below is the Deutsch expression of the word domain-user; the Windows server we used for lab practical was configured in Deutsch from the background, though the language was set to English manually.

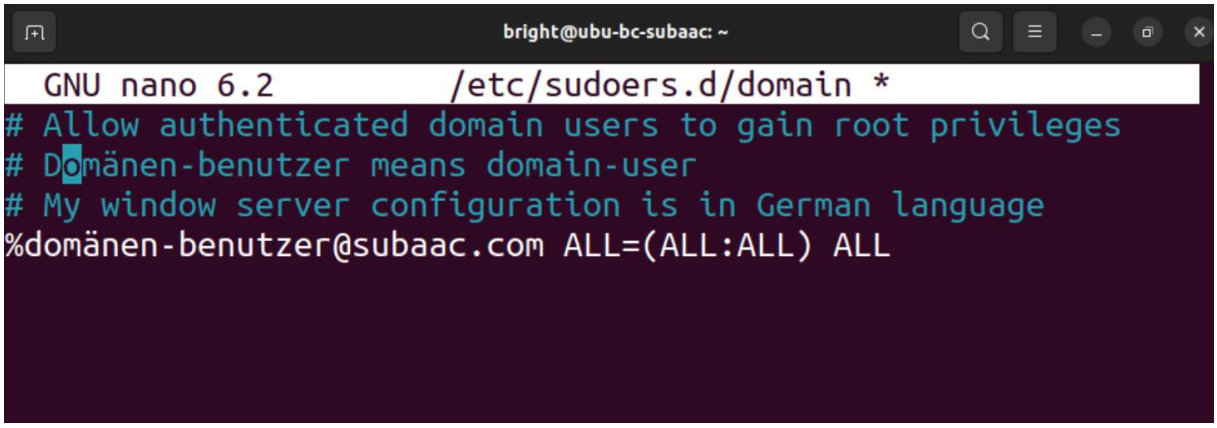


Figure 43: Configuring sudo access to domain users

To test the outcome of this configuration, as user **Sabine**, we first used the **ls** command to ensure we have a home directory for the user **Sabine**. We used the command **id** to indicate the group the user **Sabine** belongs to, which is the group name to which we granted sudo privilege; this means that every member of that group will have sudo access in the Linux environment. Finally, we used the command **sudo -l** to get complete details of the configuration outcome. The area marked green shows that our domain users are allowed full access to both the /bin and /sbin; recall that commands in /sbin are reserved for the root.

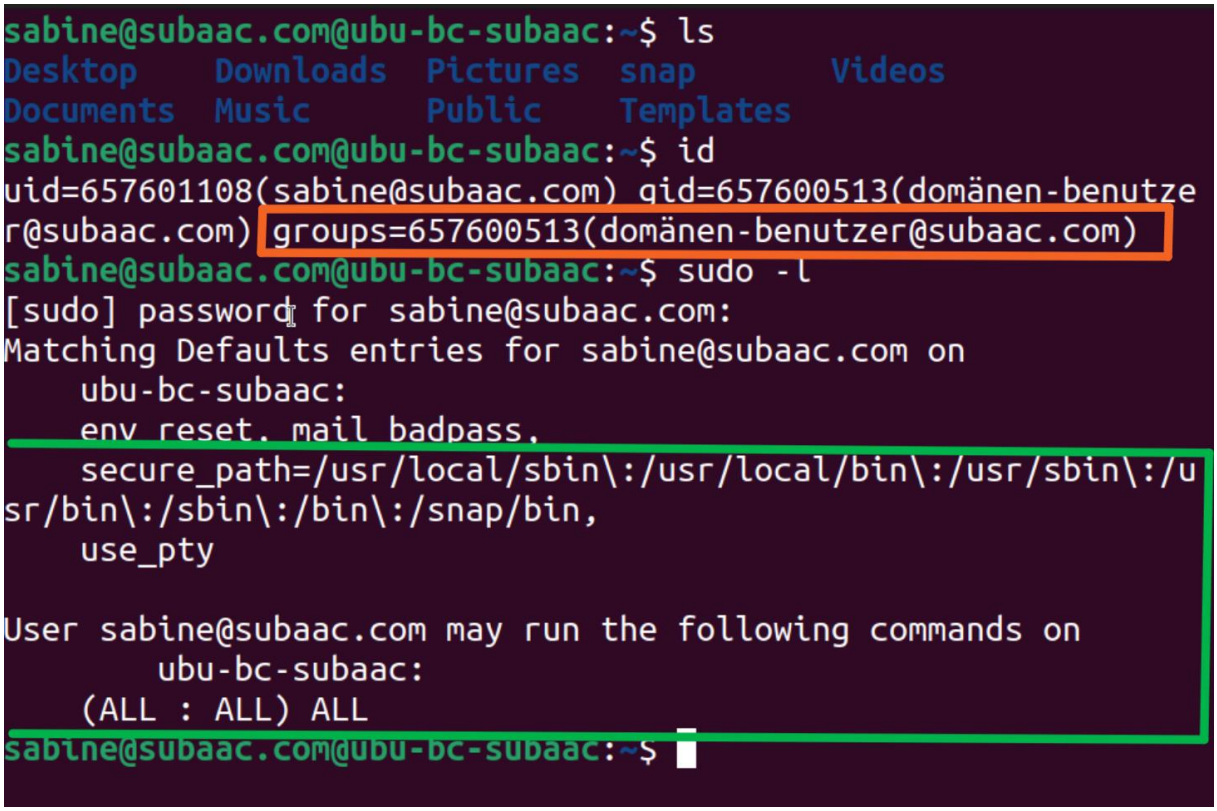


Figure 44: Confirming the sudo configuration

Lastly, before we completed the integration process, we tried to test the sudo access we gave to the domain users to ensure they could perform privileged functions on the Linux host using sudo.

```
sabine@subaac.com@ubu-bc-subaac:~$ sudo apt update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelea
se [129 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security/main am
d64 Packages [1.526 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/main i3
86 Packages [486 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main Tr
anslation-en [261 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/univers
e amd64 Packages [860 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/univers
e i386 Packages [608 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/univers
e Translation-en [167 kB]
Hit:8 http://de.archive.ubuntu.com/ubuntu jammy InRelease
Get:9 http://de.archive.ubuntu.com/ubuntu jammy-updates InRele
ase [128 kB]
Hit:10 http://de.archive.ubuntu.com/ubuntu jammy-backports InR
elease
Get:11 http://de.archive.ubuntu.com/ubuntu jammy-updates/main
amd64 Packages [1.736 kB]
Get:18 http://de.archive.ubuntu.com/ubuntu jammy-updates/unive
rse amd64 Packages [1.088 kB]
Get:19 http://de.archive.ubuntu.com/ubuntu jammy-updates/unive
rse Translation-en [251 kB]
Get:20 http://de.archive.ubuntu.com/ubuntu jammy-updates/multi
verse amd64 Packages [43,3 kB]
Fetched 11,3 MB in 5s (2.371 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
91 packages can be upgraded. Run 'apt list --upgradable' to se
e them.
sabine@subaac.com@ubu-bc-subaac:~$
```

Figure 45: The domain user Sabine is performing system updates using sudo

Before we conclude this section of this document, it is important to note that in this section, we were able to perform a live demonstration of the process involved in integrating a Linux host

into an Active Directory environment. We also demonstrated how authenticating into the Linux host can be done from the Active Directory Domain Controller. More maturely, knowing fully that every user authenticated into the Linux environment needs a default home directory and privileged permission to perform their tasks successfully. Therefore, we assigned a home directory and automated privileged permissions via sudo to authenticated users. Finally, we tested all our configurations, and all worked perfectly.

In the next chapter of this document, from an attacker's point of view, we will look deeper into the outcome of the integration we have done to discover unavoidable misconfigurations that can create attack vectors from the Linux host to the AD Domain. These newly identified attack vectors will form our results, and based on the gathered results, we will perform our risk assessment and provide recommendations, also a ground for future research.

4. Results

This chapter will deal with the negative outcome of our earlier laboratory experiment in chapter three of this document. We will analyze and detect the identifiable methods an attacker or adversary can leverage to gain access to our Active Directory environment from the integrated Linux host. This chapter is described from the perspective of a hacker or penetration tester who takes advantage of attack vectors to either gain an initial foothold or escalate privileges in a compromised environment.

We described our results in three ways. Firstly, we analyzed the configurations on the Linux host that granted us a successful integration into the Active Directory Environment; we investigated how it can help attackers fully understand our Active Directory environment. Secondly, we analyzed how it can be easier for an attacker to send network packets or crafted exploits from the Linux environment to our Active Directory Domain Controller. Lastly, we conducted internal penetration testing on our integrated Linux environment and found how Kerberos exposes the credentials of authenticated AD accounts in the Linux environment.

4.1. Exploiting Exposed Integration Configurations

In this section, to better understand why this integration configuration can threaten our Active Directory environment, we will compare Windows host integration into an Active Directory and Linux host integration into an Active Directory. However, before we go deeper into this, recall that in chapter two of this document, when we were describing these technologies, we mentioned Active Directory as a proprietary service owned by Microsoft. In contrast, we described Linux as an open-sourced Operating System built from the Unix platform; this description means that these two technologies are very far from each other and require non-proprietary, unguided, and unprotected configurations to be able to achieve a successful integration.

Windows host integration into an Active Directory Domain is a straightforward approach where an administrator will add the Windows hosts to the Active Directory network and then add it to the domain using an administrative credential. This process is completed without installing external tools or exposing configurations in any config file within the Windows environment. This is made possible because of the proprietary nature of both technologies; everything was set and controlled by the manufacturers, and there is a lot of documentation from the manufacturer about handling any security flaw that might arise from performing such integration. However, when we look closely at the integration we made in the previous chapter of this document, we will notice that the approach was not straightforward. To achieve a successful integration, we installed and configured many tools, and it is a well-known fact that **the higher the configurations, applications, processes, and services running in an environment, the higher the environment is exposed to threats.** Recall from our previous discussion in chapter two of this document about how threat actors gain an initial foothold and escalate privileges in a networked environment; it is important to know that where they cannot leverage human beings, they can leverage any vulnerability in the network.

Let us bring our discussion in this section closer to our understanding; let us use the configuration we made in Figure 38 as an example. In this document, we have already discussed broadly how attackers gain privileged permissions into the Linux environment by abusing privileges and misconfigurations to get sudo or root access. Any user or attacker with privileged access to our Linux host through any of the threats discussed earlier can easily access the sssd.conf file, and without any secondary authentication, can either tamper the setup to suit their needs or cause disruption to the system and business processes. The malicious user or attacker can also use such configuration as a point of secondary reconnaissance to understand fully how users are authenticated into the environment because a closer look at the configuration file explains thoroughly how Active Directory users authenticate and retain access in the Linux host. Apart from the configuration file we used as an example in this section, there are other files that an attacker can also study with just sudo access to perform complete reconnaissance on our Domain Controller via the integrated Linux host, looking at files in the /etc directory can expose important configurations files. Performing reconnaissance via these config files can help the attacker to proceed with the attack on the Domain Controller through the Linux host.

As discussed earlier in previous chapters, it is important to note that once a malicious user gains access to any system in a networked environment, the next step of the attack is always to look for information within that environment that can help them to gain more understanding about the full environment. An unguided and insecure configuration can grant a malicious user an internal reconnaissance to gather much information they can use to achieve privilege escalation or move laterally within other systems connected to the environment to achieve their actions and objectives.

4.2. Exploiting The Domain Controller from the Linux Host Using Malicious Payloads

Malicious payloads can be lines of code or scripts crafted to perform malicious activities or compromise an environment to gain unwanted access. They can also be written as exploits or embedded into network packets. Sometimes, they can be embedded into emails to perform fishing attacks, but phishing attacks are out of scope for this section. This section will depend more on exploit codes and network packet payloads to describe the second attack vector we noticed during our integration process.

Exploit code is a crafted payload that can be delivered to a victim through a vulnerable service or protocol. A network packet sends data across systems within a network or to another network outside the system's geographical network.

Recall that in the previous chapters and sections, we have mentioned that once a malicious user, attacker, or adversary gains access to a system within a networked environment, the next stage is looking for ways to compromise other systems because most times, the first system that grants the access might not be a major target. The case study in this document is not an exemption. Once access is gained on the Linux host, next is looking for a way to move into the AD environment. Let us bring this section closer to our understanding. Recall that we

sent different network packets to the Domain Controller during integration. According to Figures 30 and 28, we sent an ICMP echo request and an Nmap request to the Domain Controller, the Domain Controller received our requests and replied to them accordingly. Figure 33 also mapped the Linux host's DNS server to the Domain Controller's DNS server. These processes are very necessary for a successful integration because for the Linux host to be seen as an object in the Active Directory environment entails that there must be unrestricted communications between the two technologies, and the Linux host must point some of its features to the Domain Controller. An attacker who has performed internal reconnaissance on the Linux machine and discovered details about the Active Directory environment can use the discovered network and DNS information to craft malicious payloads in the form of packets and send them to the AD Domain Controller.

Secondly, recall the output of the Nmap packet we sent to the AD Domain Controller in Figure 28. This scan returned all the opened ports and services running on the AD Domain Controller. In real-world hacking or penetration testing, this is one of the network scans done on victims' hostnames or IP addresses to detect vulnerable ports and services opened on the victim's system. We will not go deeper into Nmap and how the technology functions because it is out of the scope of this work. Furthermore, knowing that such scans can reveal hidden information to attackers when used properly is very essential. Just as the result we got in Figure 28, an attacker can leverage this result to go deep to perform more research on those services, check for their expected vulnerabilities, and craft or leverage commercial exploits as weapons and deliver them to the AD Domain Controller via the vulnerable port or service. This procedure follows the process we discussed during our adversary emulation using the **Cyber Kill Chain** and **MITRE ATT&CK** in section 2.3.2. of this document.

This second attack vector we discovered is another critical vector that an attacker can use to perform more reconnaissance and send crafted malicious payload directly to our AD Domain Controller as a network packet or exploit.

4.3. Reusing Exposed Kerberos Authentication Ticket

Earlier in section 2.2.4., we discussed how Kerberos authenticates users before granting them access to any resources or objects in the Active Directory environment. We explained that any user who requires access to any resource in the AD environment will begin the Kerberos authentication process. The user will send an Authentication Server Request (AS-REQ) message to the Domain Controller (DC). This will automatically encrypt the timestamp of that message request with the user's hashed password and username. the Domain Controller receives the request and attempts to decrypt it with the user's password hash record stored in the **ntds.dit** file, if this is successful, the Domain Controller will send back an Authentication Server Response (AS-REP) message; this message will contain the user's session key encrypted by the user's password and a Ticket Granting Ticket (TGT) issued by the Key Distribution Center (KDC). Once the user receives this response, the server considers the client authentication complete.

The explanation above is no exception when a user wants access to the Linux host that is an object in the active directory environment. Understand that the outcome of the practical integration we did in chapter three entails that our Linux host is now an object in the Active Directory environment and must be accessed just like every other object or resource within the domain. Therefore, the process stated in the earlier paragraph is also automated for any user who wants to authenticate to the Linux host using their AD credentials. Such users must have a Ticket Granting Ticket (TGT), encrypted with their password and session key, to access the Linux host.

Recall in our sssd.conf file in Figure 38, we set cache credentials to be equal to true. This setting is important because it allows logins even when the AD Domain Controller is unreachable. This is a method of business continuity where users and administrators can still log in and perform their tasks on the Linux host even if the AD Domain Controller is not reachable. Using our root access on the Linux host as an attacker, we conducted internal penetration testing by navigating around the important directories in search of sensitive information to escalate privileges. During this search, we noticed that there is a Kerberos file attached to an ID number in the **/tmp** director of the Linux host. The file looks suspicious because of the name, given that Krb5 is one of the tools known for authentication reasons.

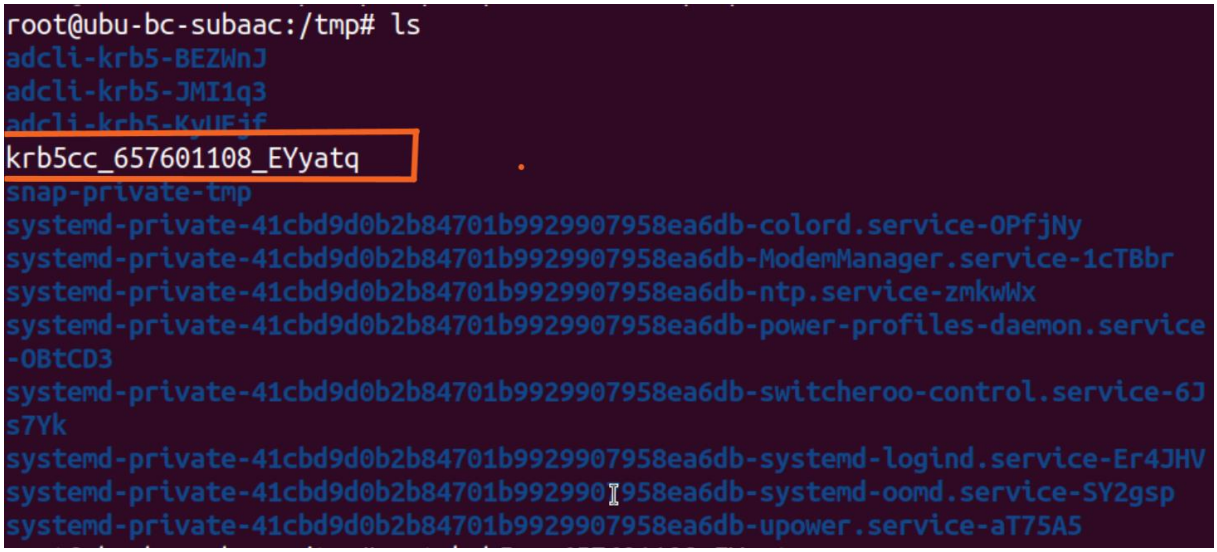


Figure 46: Kerberos file found in the /tmp directory in the Linux host

Out of curiosity, which comes with the hacker's mindset as discussed in the adversary emulation in chapter two, we used the Linux **cat** command to list the file's content; we noticed that it contained the TGT for the user **Sabine**, who used her AD credentials to log in to the Linux host. This TGT is saved there so the user can still reuse it to have access in case the Domain Controller is not reachable. The file includes our AD Domain controller's hostname and the ticket owner's name. The remaining part of the ticket is a binary file containing some elements, as mentioned earlier in this section; TGT is encrypted alongside the user's password and session key so the user can always reopen a session using the TGT.

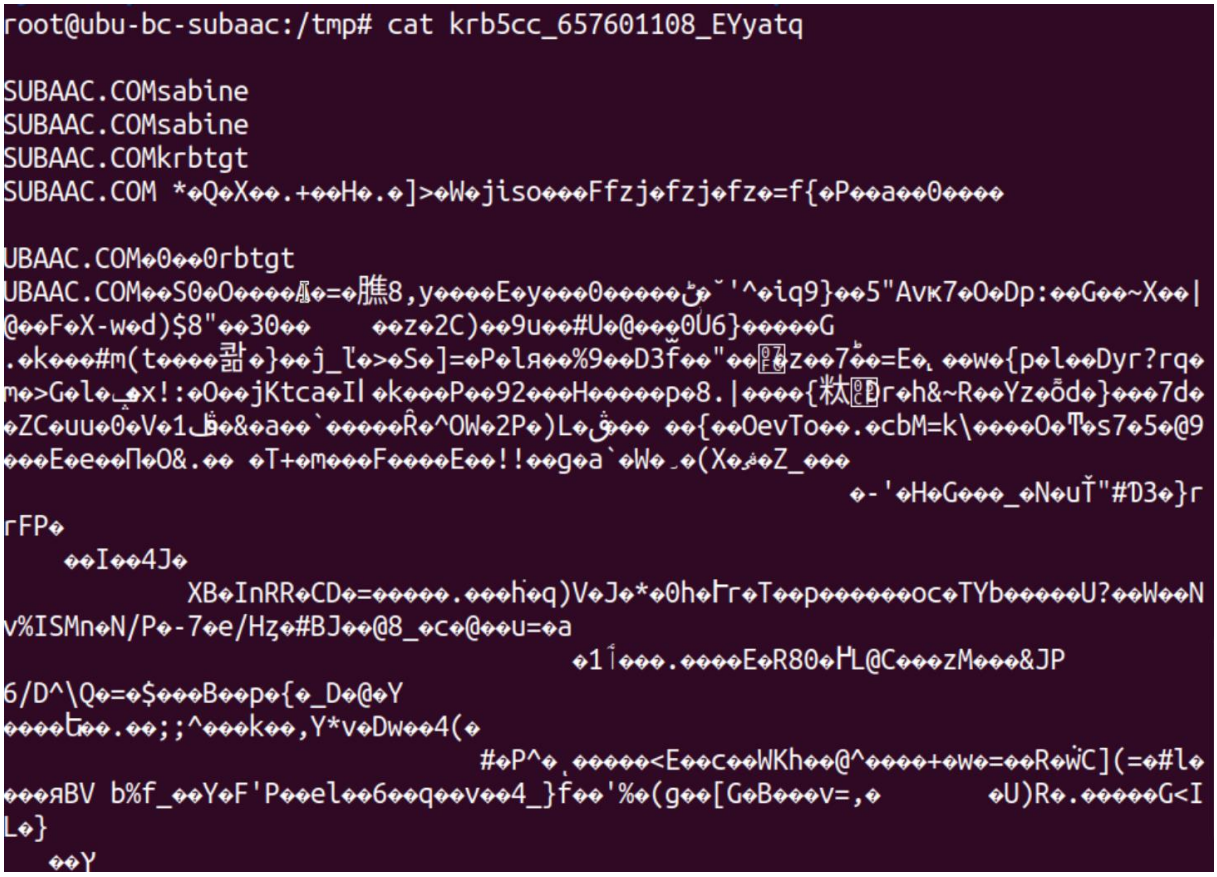


Figure 47: The TGT for user Sabine

More research on the reuse of TGT by an attacker shows possible outcomes that an attacker can reuse the ticket through the following ways: Present the stolen ticket to have access to the object or resources for which the ticket is valid, move laterally within the network, access other systems and resources as the compromised user, attempt privilege escalations if the captured TGT belong to a high privileged user, it can also be used to have access to shares made by the compromised user. These TGT attacks can be achieved through the **Pass the Ticket attack (**Daniel Petri. 2024). One dangerous and challenging thing about a Pass the Ticket attack is its detection. This is because the attackers use legitimate tickets. Therefore, the actions often appear authentic, just like an authorized user's activities.

In this section and previous sections of this chapter, we have exposed three different threats and attack vectors that attackers can use to gain a foothold or escalate privileges in the Active Directory environment via the integrated Linux host. In the next section of this chapter, we will perform a threat evaluation, also known as risk assessment, on the newly discovered attack vectors (threats).

4.4. Threat Evaluation

In this section, we will perform a risk assessment on our newly discovered attack vectors. We will evaluate their likelihood and impacts while suggesting possible mitigations for each threat. To achieve this evaluation, we will use the OWASP Risk Rating Calculator to automate this process.

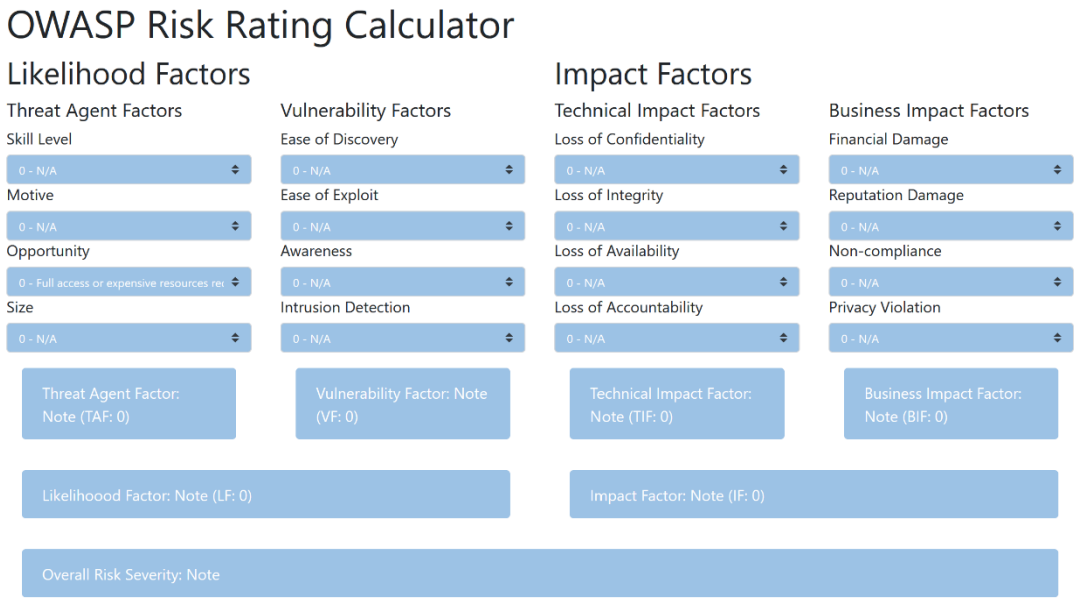


Figure 48: OWASP Risk Rating Calculator (OWASP)

The OWASP Risk Rating Calculator automates the calculation of the likelihood and impact of an attack using some factors and methodologies, as seen in the image above. The description of its technicality is outside the scope of this document. More research can be done to understand how it works.

To achieve our aim in this section, we will prepare a table for threat evaluation, list the threats one after the other, and input their likelihoods and impacts based on the factors listed in Figure 48 above. Finally, we suggest mitigation strategies for the discovered threats.

S/N	Threats	Likelihood	Impact	Mitigations
1.	Exploiting exposed integration configurations.	High	High	Restrict the number of privileged AD users and groups in the Linux host. If an application is hosted on a Linux host, strict hardenings are recommended, as

				mentioned in the OWASP Top 10 Application Security. Constantly monitor users' activities on the Linux host. Regularly backup configuration files and test recovery procedures via stored backups to ascertain data integrity and availability. Disable remote root ssh login if it is not necessary.
2.	Exploiting the Domain Controller from the Linux host using malicious payloads.	Medium	High	Block unneeded traffic, ports, and scans that can be used to send payloads from the Linux host to the AD Domain. Remove unneeded services, Constantly Patch, and update all needed services in the AD Domain. Constant Log Monitoring. Presence of IPS and IDS systems
3.	Reusing exposed Kerberos authentication ticket.	Medium	High	Encrypt all communications between the Linux and AD using TSL/SSL. Enable multi-factor authentication, two-step verification, or certificate-based authentication. Perform regular Penetration Testing. Instead of Password Authentication, use SSH key-based authentication.

				Involve centralized monitoring tools like SIEM and include Kerberos-related Logs. Reduce ticket lifetime to limit the ability of ticket reuse. Enable screen locks to ensure the user's sessions are locked when unused.

Table 2: Threat evaluation table

Explanation of Likelihoods and Impacts for the Above Evaluated Threats:

- **Exploiting Exposed Integration Configurations**: The likelihood is high because the threat actors do not require high skills to perform internal reconnaissance on a compromised system, and having root access to the Linux host can grant such users easy access to files and processes in the environment. The motive is very high because internal reconnaissance is always the next approach once an environment is compromised, and the attacker will need more information for lateral movement. The impact is also high because apart from obtaining sensitive information from the configuration file, the attacker might choose to tamper with the configuration files, resulting in a Denial-of-Service attack and breach communications between the Linux host and AD Domain controller.
- **Exploiting The Domain Controller from the Linux host via Malicious Payloads:** The likelihood is medium because while the motive is high, it also requires high-level skills in attacking or pen testing to obtain a root shell in the AD Domain via this threat. Sometimes, attackers might have to leverage commercial exploits to achieve their desired objective. This requires high skills and capital to finance the purchase of an exploit or payload, and in most cases, it takes much time to achieve and might be detected if not carefully done. While the likelihood is medium, the impact is high because the success of such an attack can grant a root shell in the AD Domain, allowing the attacker to take full control of the domain.
- **Reusing Exposed Kerberos Authentication Ticket:** Just as the second threat we analyzed earlier, this also has a medium likelihood because both have the same methodology. Performing a Pass the Ticket attack using a valid TGT requires high attacking skills. Though the motive is always high and sometimes does not require much capital to perform the attack. However, it requires great skills and is sometimes time-consuming because of its technicality. It can be detected with strong log monitoring and implementation of IDS systems. While the likelihood is medium, its impact is high because such an attack can be used to impersonate a user and escalate privileges in the AD Domain.

5. Discussion and Recommendation

Cybersecurity is a journey, not a destination (Mathieu Gorge, 2021). The earlier clause explains that learning should never stop in a continuously evolving industry like cybersecurity. Attackers, threat actors, adversaries, etc., build their knowledge on top of the knowledge of the defense team. Therefore, while you think your defense team has done their best, an attacker is somewhere thinking of how to manipulate that best to make it worse. The 2021 Colonial Pipeline Company attack is a very good example to understand the message we are passing on in this paragraph. This attack happened between April 29th - May 6th, 2021. The attackers stole and locked up to 100 gigabytes worth of company data, including the personal information of nearly 6,000 individuals. The hackers demanded a ransom of 75 bitcoins (worth about $4.4 million at the time) to be paid by the company before they could receive the key to unlock the data. During the post-incidence activities, it was discovered that the root cause of the action was a legacy virtual private network (VPN) account that allowed employees to access the company's network remotely. It was also noted that the account, though no longer in use (legacy), was still there, serving as an endpoint in the company's network. The VPN account's password was found inside a batch of leaked passwords on the dark web, so this could be that a colonial employee may have used the same password on another non-company website that was previously hacked (Suraj Srinivasan & Li-Kuan (Jason)Ni, March 2023). Before this incident, Colonial increased its IT budget by 50% and spent more than $200 million on its IT systems between 2016 and 2021, on top of over $1.5 billion for the physical integrity of the pipeline. During the ransomware attack in May 2021, the company's IT network was strictly segregated from the pipeline operating control systems and had active monitoring and overlapping threat-detection systems. In particular, at least three different software tools were in place that would provide alerts when data left the network. The expanded cybersecurity regime also included regular simulated phishing campaigns for employees. The CEO also testified that the company took cybersecurity **extremely seriously** and claimed that the board of directors had never denied Mouchet's request for cybersecurity funding (Suraj Srinivasan & Li-Kuan (Jason)Ni, March 2023).

This attack explains that while Colonial Pipeline Company had already concluded that they had done their best and had the best cybersecurity posture, the attacker could turn those best to worse. The breach was done through an overlooked legacy VPN account, maybe because nothing could happen through such an endpoint. The account password was captured from a breach on another website and was used to gain access to the Colonial Pipeline Company's network. These are common things we neglect, but comparing this attack with the frameworks we discussed in this document, we should understand how far an attacker goes to have unauthorized access into our environment and perform their actions and objectives.

Most common and recent attacks on Active Directories follow the same procedure described above. Most of the time, it is not about the Active Directory as a major server and hacker's major target; it is about the endpoints that can grant attackers privileged access to the AD environment. In my experience as a System Administrator at SimScale Engineering Simulation

Company in Germany, I have noticed that the endpoint to the active directory environment is not only the network access but includes all the objects in the environment, such as computers, user accounts, group accounts, protocols, and other technologies integrated into the AD environment. Therefore, in my company, even before integrating any third-party tool into our AD environment, we must be sure that the third-party vendor prioritizes cybersecurity and provides proof of that by signing a compliance agreement with us.

In this document, we have described the integration of Linux technology into our AD environment. Looking closely at the image in Figure 36, we should notice that our Linux host is now an object in our AD environment. Any attack on it can also grant access to the AD environment; this access could be privileged in some cases. For example, we got the TGT for the user Sabine@subaac.com. Let us log in to the AD environment to check the user's permissions.

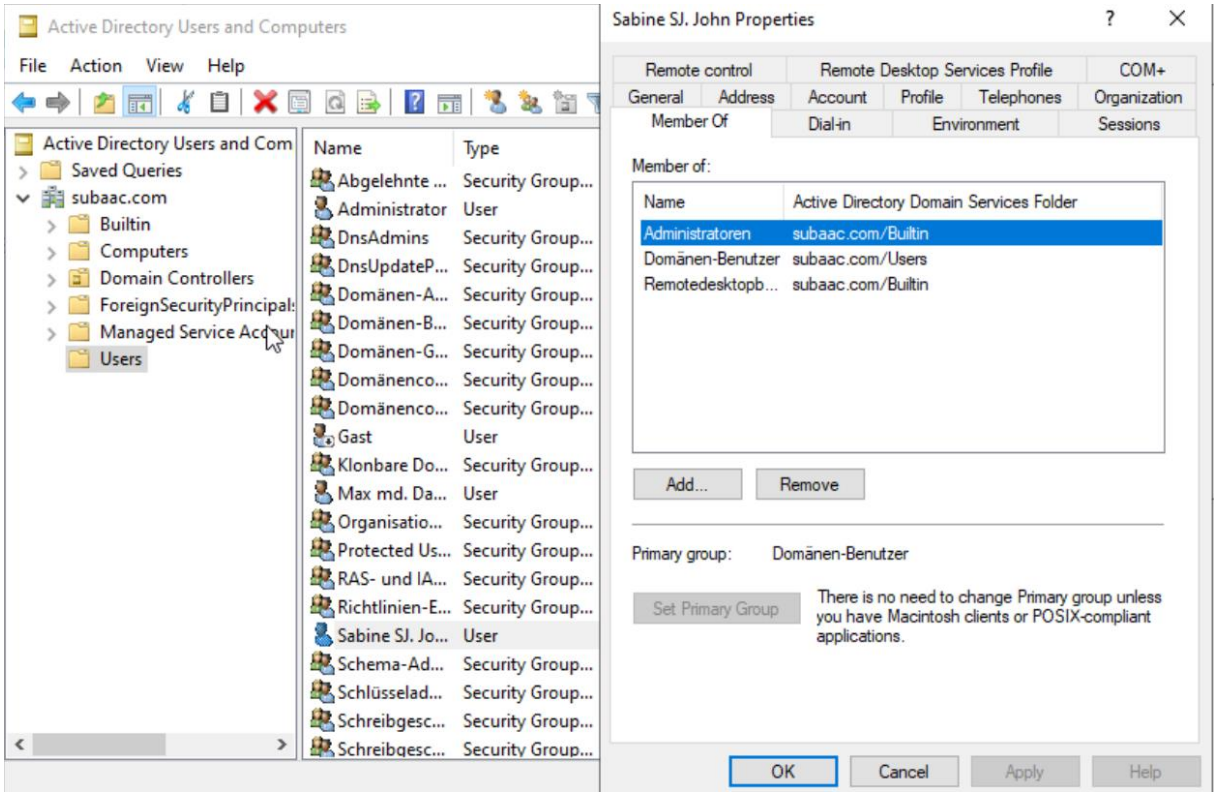


Figure 49: User Sabine's permissions in the AD environment

A closer look at the image above shows that **Sabine** is an administrative user in the AD environment. This means the TGT we retrieved earlier can grant us administrative access to the Domain Controller.

While hosting Linux in an AD environment, understand that Linux is not a proprietary service, as mentioned earlier in this document. This means that no Linux vendor can guarantee security assurance, provide security updates and patches for users, or even sign any compliance agreement with users. Updates on Linux are done based on research of known vulnerabilities.

Therefore, in most cases, Linux users use it at their own risk. Following this understanding, we recommend that while considering whether to integrate your Linux host into your company's AD environment, you may need to consider these three important considerations.

- **Consider the Necessity:** As a Chief Executive Officer (CEO) thinking of such integration, ask yourself this question: Is it necessary to do this? In chapter three of this document, we highlighted the situations that must give rise to considering this integration as necessary. If not for those reasons, there should be no need for that. Therefore, I recommend this document to every CEO who is thinking about whether to perform such integration or not.
- **The Risk:** As a Chief Information Security Officer (CISO), your CEO approaches you to discuss the idea and the importance of integrating the organization's Linux hosts into the AD environment for an improved business process. What risks would you mention to the CEO, what are their likelihoods and impacts, and what suggested mitigation strategies can you give to avoid disruption arising from such integration? Therefore, I recommend this document to all CISOs brainstorming on having a good security posture while still doing this integration.
- **Access Right:** As a System Administrator, your CISO may ask you to allow some AD accounts to access the Linux host to facilitate their job. How will you initiate this account access without abusing privileged rights and permissions? Therefore, I also recommend this document to system administrators who are considering how to get robust system usage while managing access between the Linux host and the AD domain.

This document is generally recommended to all CEOs, CISOs, System Administrators, Linux and Active Directory users, and the public, whose decision-making would be aided by the information contained in this document.

6. Conclusion and Future Work

6.1. Conclusion

In this document, we answered the following research questions: *Why do organizations that use Linux hosts always see integrating them into their AD environment as a better option? What types of threats are these organizations exposed to because of this integration? What suggested mitigations can be in place to mitigate these threats?*

We experimented with the integration procedure in our laboratory environment to achieve a perfect and realistic result. During our experiment, we followed the current enterprise integration method and used the Linux distribution currently used by companies to ensure we used a system that matches the evolving technology. This makes this document stand to the taste of time.

During the integration, we noticed two already visible threats: **exposed integration configurations** and exploitation that **can be done via malicious payload due to open communication channels between both environments.** After our integration, we also conducted a penetration testing exercise on the AD environment through the integrated Linux host. We performed internal reconnaissance on the Linux host to see if we could find information that we could use to get an initial foothold or to escalate privilege in the AD environment. During this exercise, we detected our third threat vector, the **exposed Kerberos authentication ticket.** We discovered these threat vectors and explained how attackers can take advantage of them to cause harm in our AD environment. We performed a threat evaluation and risk assessment on this newly discovered threat to understand their exploitation's likelihood and impact. Finally, we suggested mitigation strategies that can help to limit both the likelihood and impact of their exploitation.

Broadly, we can conclude that companies need to bring their platforms and infrastructures together for easy monitoring and management. However, knowing the risks involved in such decisions is also essential. Prioritizing these risks and looking for a way to mitigate them is crucial before implementing the decisions. Even after mitigation, constant checking and monitoring, and penetration testing should be in place because, as we mentioned earlier in this document, the best defense is a good offense, meaning that being proactive is an excellent defensive skill in cyber security.

6.2. Future Work

We used the Ubuntu Linux distribution during our laboratory experiments. We chose that to meet enterprise demands based on usage statistics. Unfortunately, because of limitations and time constraints, we could not experiment with other Linux distributions to see if we would achieve a different result. RedHat also tried to experiment with RedHat Linux Enterprise, which seems to be the same outcome, though RedHat did not emphasize more on hackers' perspective as we did in this document (Delehaye et al., 2024). This calls for future work in this area

to look out for other Linux distributions and integrate them into our AD environment, focusing on hackers' perspective to see if it will produce the same results or react differently.

Secondly, because this is a time-based project, we were trapped in using specific types of tools to perform the integration experiment that we did in this document. Refer to Figure 31 for the tools we used to integrate these technologies successfully. We also took time to explain all these tools and what contributions each made towards successful integration. Either now or in the future time, there could be other tools, scripts, and applications to perform this process. More research can still be done about them, and they can also be engaged in repeating the experiment we did in this document to find out if they can produce a different result judging from the hacker's point of view.

7. References

Archiveddocs. (2014, November 19). *Active Directory Collection: Active Directory*. Microsoft Learn. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780036(v=ws.10)?redirectedfrom=MSDN.

Berkouwer, S. (2022). Active Directory Administration Cookbook (second edition). URL: https://github.com/PacktPublishing/Active-Directory-Administration-Cookbook-Second-Edition. ISBN-13: 9781789806984.

Boldrito, R. & Esteve, J. (2007 & 2009 edition). GNU/LINUX Advanced Administration. URL: https://ia804501.us.archive.org/21/items/GnuLinuxAdvancedAdministration/Gnu%20Linux%20advanced%20administration.pdf. ISBN-10: 0201596075. ISBN-13: 978-0201596076.

Bossiroy, B. (2023). Most common Active Directory misconfigurations and default settings that put your organization at risk. URL: https://blog.nviso.eu/2023/10/26/most-common-active-directory-misconfigurations-and-default-settings-that-put-your-organization-at-risk/.

Bourbita, T. (2023). [Comment on the blog post *How to enabled all those users who have disabled kerberos preauthentication. - Microsoft Q&A*. (n.d.).] URL: https://learn.microsoft.com/en-us/answers/questions/1192530/how-to-enabled-all-those-users-who-have-disabled-k.

Boutnaru, S. Ph.D. (2022). Linux Security — Secure Computing Mode (seccomp). URL: https://medium.com/@boutnaru/linux-security-secure-computing-mode-seccomp-2314c3e77a76.

Bovet, D. P., & Cesati, M. (2006). *Understanding the Linux kernel*. O'Reilly Media, Inc. URL: https://www.cs.utexas.edu/~rossbach/cs380p/papers/ulk3.pdf. pp. 1 – 187 & pp. 398 – 419. pp. 809 – 829. ISBN-10: 0-596-00565-2. ISBN-13: 978-0-596-00565-8.

Branka. (February 2024). Linux Statisticts-2024. URL: https://truelist.co/blog/linux-statistics/.

Brian, D., Richard, J., Allen, R., & Lowe-Norrs, A. (April 2013). Active Directory 5th Edition. URL: https://www.academia.edu/43517791/Active_Directory_5th_edition. pp.1 – 61 ISBN: 978-1-449-32002-7.

Chua, I. (2022). Real life Examples of Web Vulnerabilities. OWASP Top 10. URL: https://www.horangi.com/blog/real-life-examples-of-web-vulnerabilities.

Cobbaut, P., Serge van Ginderachter, Hendrik De Vloed, Wouter Verhelst, Geert Goossens, Elie De Brauwer, Christophe Vandeplas, Bert Desmet, & Rich Yonts. (2015). *Linux Networking*. URL: http://linux-training.be/linuxnet.pdf. 294 pages. ISBN-10: 9888406213. ISBN-13: 978-9888406210.

Cobbaut, P., Serge van Ginderachter, Ywein Van den Brande, Hendrik De Vloed, Wouter Verhelst, Geert Goossens, Elie De Brauwer, Christophe Vandeplas, Bert Desmet, & Rich Yonts. (2015). *Linux Fundamentals*. URL: https://linux-training.be/linuxfun.pdf. pp. 6 – 35. pp. 59 – 135. pp. 142 – 220. pp. 266 – 336. ISBN-10: 9888406167. ISBN-13: 978-9888406166.

Delehaye, F., Red Hat Customer Content Services, Muehlfeld, M., Hanzelka, F., Maňásková, L., Šteflová Petrová, A., Čapek, T., & Ballard, E. D. (2024). *Red Hat Enterprise Linux 7 Windows Integration Guide*. https://docs.redhat.com/en-us/documentation/red_hat_enterprise_linux/7/pdf/windows_integration_guide/Red_Hat_Enterprise_Linux-7-Windows_Integration_Guide-en-US.pdf.

Garrels, M. (2008). *Introduction to Linux* (1.27). URL: https://tldp.org/LDP/intro-linux/intro-linux.pdf. pp. 7 – 155.

Gorge, M. (2021). Cybersecurity is a journey, not a destination. URL: https://www.forbes.com/sites/forbesbooksauthors/2021/10/11/cybersecurity-is-a-journey-not-a-destination/.

GTFOBins. An organized list of Unix binaries that can be used to bypass security and misconfigured applications in Unix systems GTFOBins Page. Retrieved July 12, 2024, from https://gtfobins.github.io/.

Karapetyants, N., & Efanov, D. (2022). A practical approach to learning Linux vulnerabilities. *Journal of Computer Virology and Hacking Techniques*, *19*(3), 409–418. https://doi.org/10.1007/s11416-022-00455-w.

Kost, E. (2021). Cyber Threat. URL: https://www.upguard.com/glossary/cyber-threat.

Linux Audit. Linux and ASLR: kernel/randomize_va_space. Retrieved May 12, 2024, from URL: https://linux-audit.com/linux-aslr-and-kernelrandomize_va_space-setting/.

Lockheed, Martin. T Cyber Kill Chain®. Retrieved May 25, 2024, from https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

Marchant, G. (2023). Official CompTIA CS0-003 Study Guide (Exam CS0-0003). URL: https://www.comptia.org/. ISBN-10: 1642744840 ISBN-13: 978-1642744842

Market Share of Microsoft Active Directory. 6sense Page. Retrieved April 22, 2024, from https://6sense.com/tech/identity-and-access-management/microsoft-active-directory-market-share.

Mauerer, W. (2008). *Professional Linux Kernel Architecture*. http://ci.nii.ac.jp/ncid/BA90926786. pp. 733 – 892. ISBN-10: 0470343435. ISBN-13: 978-0470343432.

Microsoft. (2022). Active Directory Service Overview. URL: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

Microsoft. (2023). Active Directory Security Groups. URL: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-groups.

MITRE ATT&CK Linux Matrix. 2024. URL: https://attack.mitre.org/matrices/enterprise/linux/.

MITRE ATT&CK Widows Matrix. 2024. URL: https://attack.mitre.org/matrices/enterprise/windows/.

MITRE ATT&CK. 2024. URL: https://attack.mitre.org/matrices/enterprise/.

MITRE. 2024. URL: https://www.mitre.org/.

Moser, M. (2023). System and Network Security. iptables firewalls WS 23/24. [PowerPoint slides]. Hochschule der Bayerischen Wirtschaft (HDBW). URL: https://moodle01.hdbw-hochschule.de/pluginfile.php/71688/mod_resource/content/1/Firewalls-Part-2-IPTables.pdf.

Offsec. (2023). Penetration testing with Kali (Pen - 200). URL: https://portal.offsec.com/courses/pen-200/books-and-videos/modules.

OWASP Risk Rating Calculator. URL: https://owasp-risk-rating.com/.

OWASP Top 10. (2021). URL: https://owasp.org/Top10/

Packet Labs. (2023). 8 Best Practice for Active Directory Security. URL: https://www.packetlabs.net/posts/best-practices-for-active-directory-security/.

Perishable. (2023, August 27). *Demystifying Active Directory: domains, forests, and key concepts*. Paddy Maddy. https://www.paddymaddy.com/demystifying-active-directory-domains-forests-and-key-concepts/

Perla, E., & Oldani, M. (2011). *A Guide to Kernel exploitation*. Elsevier Inc. URL: https://paper.bobylive.com/Security/A%20Guide%20to%20Kernel%20Exploitation%20%20Attacking%20the%20Core.pdf. pp. 1 – 99. pp. 343 – 384 ISBN 978-1-59749-486-1 (pbk. : alk. Paper).

Petri, D. (2024). How To Defend Against Pass the Ticket Attack: AD Security 001. URL: https://www.semperis.com/blog/how-to-defend-against-pass-the-ticket-attack/.

Ross, A. (January 2017). The Ultimate Linux Newbie Guide. URL: www.linuxnewbieguide.org.

Sakari, K. & Mika, S. (2002). Inside Active Directory 2nd Edition. URL: https://books.google.co.bw/books?id=36t7zE8VTeAC&printsec=frontcover#v=onepage&q&f=false. pp. 1 – 117. pp. 202 – 306. pp. 419 – 508. ISBN: 0-201-61621-1.

Sengupta S. (2020). How To Harden Your Docker Containers Using Seccomp Security Profile. URL: https://hackernoon.com/how-to-harden-your-docker-containers-using-seccomp-security-profile-81153ucz.

Shameli-Sendi, A. (2021). Understanding Linux kernel vulnerabilities. *Journal of Computer Virology and Hacking Techniques*, *17*(4), 265–278. pp. 1 – 4. https://doi.org/10.1007/s11416-021-00379-x.

Shotts, W. (2021). Adventures with the Linux Command Line 1st Edition. URL: https://unlimited.dl.sourceforge.net/project/linuxcommand/AWTLCL/21.10/AWTLCL-21.10.pdf?viasf=1. Pp. 49 – 218. ISBN-10: 9781593273897. ISBN-13: 978-1593273897.

Simmons, C. & IDG Books Worldwide, Inc. (2001). *Active Directory Bible*. IDG Books Worldwide. URL: https://www.uv.es/martineu/ad/bibliaAD.pdf. pp. 1 – 106. ISBN: 0-7645-4762-3.

Spreitzenbarth, M. (2024). Crises Readiness. [PowerPoint slides]. Hochschule der Bayerischen Wirtschaft (HDBW) URL: https://moodle01.hdbw-hochschule.de/pluginfile.php/74930/mod_resource/content/1/Crisis%20Readiness%20Slides2024.pdf.

Srinivasan, S. & Li-kuan, (Jason)Ni. (March 2023). Harvard Business School. Ransomware Attack on Colonial Pipeline Company. URL: https://moodle01.hdbw-hochschule.de/pluginfile.php/74957/mod_assign/introattachment/0/Case_Study_HDBW.pdf?forcedownload=1.

Stanek, W. (2009). Active Directory Administrator's Pocket Consultant. URL: https://ptgmedia.pearsoncmg.com/images/9780735626485/samplepages/9780735626485.pdf. ISBN-10: 0735626480. ISBN-13: 978-0735626485.

Suehring, S. (January 2015). Linux® Firewalls. Enhancing Security with nftables and beyond. Fourth Edition. URL: https://bmansoori.ir/book/Linux_Firewalls_Enhancing_Security.pdf. pp. 1 – 261. ISBN 978-0-13-400002-2 (pbk. : alk. paper)—ISBN 0-13-400002-1 (pbk. : alk. paper).

Vaidyanathan, R. (2024). Defending Against Active Directory Attacks. URL: https://download.manageengine.com/log-management/ebooks/active-directory-for-dummies-ebook.pdf. 40 pages. ISBN 978-1-394-20795-4 (pbk); ISBN 978-1-394-20796-1 (ebk).

Zomaya, D. (June 2023). Linux File Permissions: Understanding setuid, setgid, and the Sticky Bit. URL: https://www.cbtnuggets.com/blog/technology/system-admin/linux-file-permissions-understanding-setuid-setgid-and-the-sticky-bit.