

# Fundamentals of Information System Security SEC285



PROFESSOR ERIC AGHADIUNO


By Hector Acosta



# Introduction:

In today's digital world, cybersecurity is more important than ever. Protecting sensitive data and preventing unauthorized access are crucial for any organization. In this project, I worked on securing a Linux server by using various cybersecurity techniques. This included encrypting data, setting up a firewall, creating a security policy for personal devices (BYOD), implementing multi-factor authentication (MFA), and conducting vulnerability assessments. The goal of this project was to demonstrate how these different security methods can be applied to protect a system from potential threats and improve overall security.





# Asymmetric Encryption & Secure File Management

## **Introduction:**

In this module, I worked on securing files using asymmetric encryption with GPG (GNU Privacy Guard).

Encryption ensures that sensitive information is protected from unauthorized access.

Additionally, I used the shred command to permanently delete files, enhancing security by preventing data recovery.

The goal was to demonstrate how to encrypt, decrypt, and securely manage files in a Linux environment.

# File Encryption

## Overview:

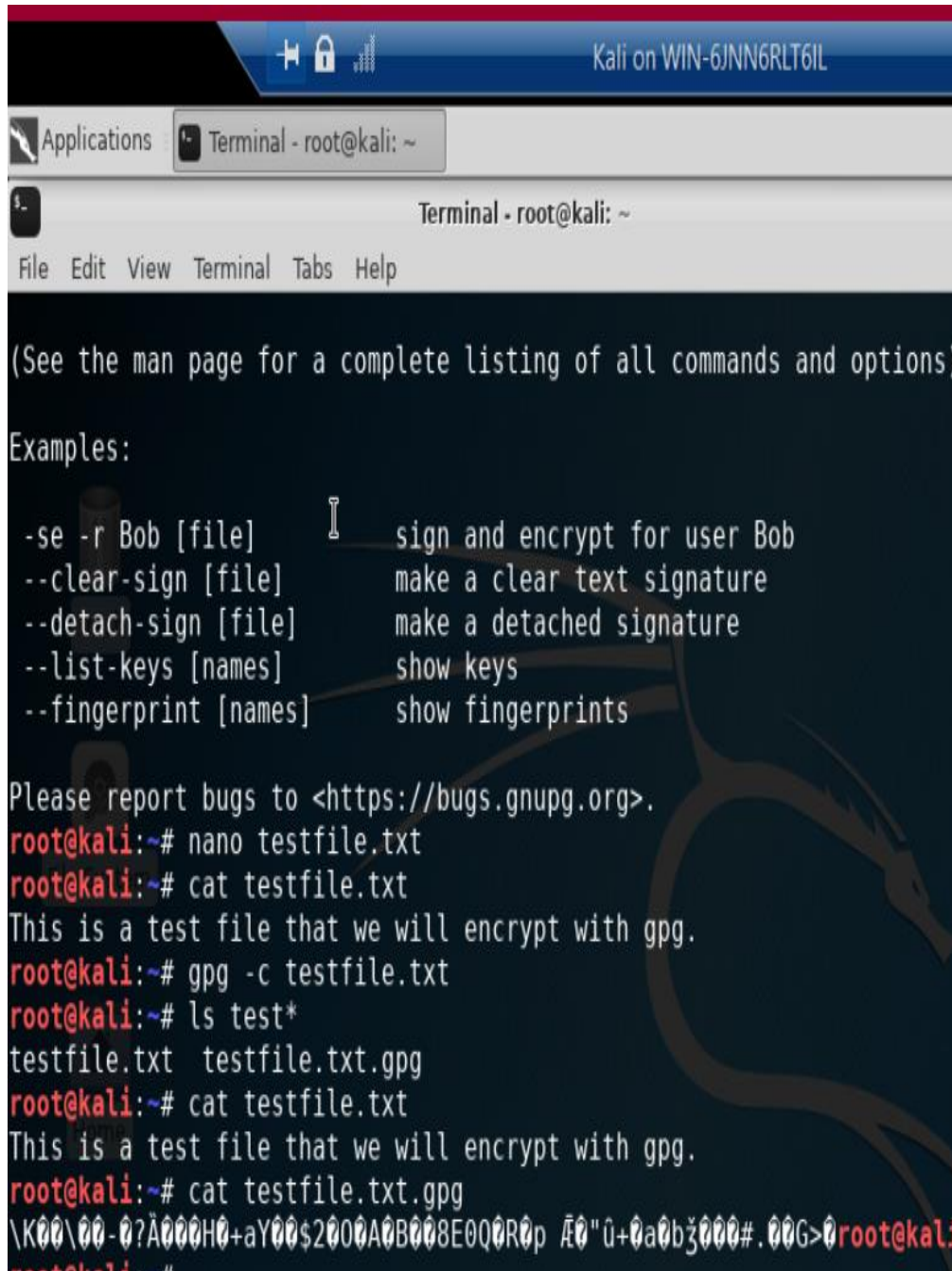
- The screenshot demonstrates the process of encrypting a plaintext file using GPG in the Kali Linux environment.

## What's Shown:

- **Content of the Plaintext File:**
  - The command `cat testfile.txt` was used to display the content of the original plaintext file.
  - The file contains the sentence: *"This is a test file that we will encrypt with gpg."*
- **Encrypting the File:**
  - The command `gpg -c testfile.txt` was used to encrypt the plaintext file.
  - After encryption, the `ls` command shows both the original file (`testfile.txt`) and the newly created encrypted file (`testfile.txt.gpg`).
- **Verifying the Encrypted File:**
  - To confirm the encryption, the `cat testfile.txt.gpg` command was used. As expected, the content of the encrypted file is unreadable, ensuring the security of the data.

**Result:**

- The encryption process was successful, with the encrypted file (testfile.txt.gpg) created and its contents properly secured.





# File Decryption

## Overview:

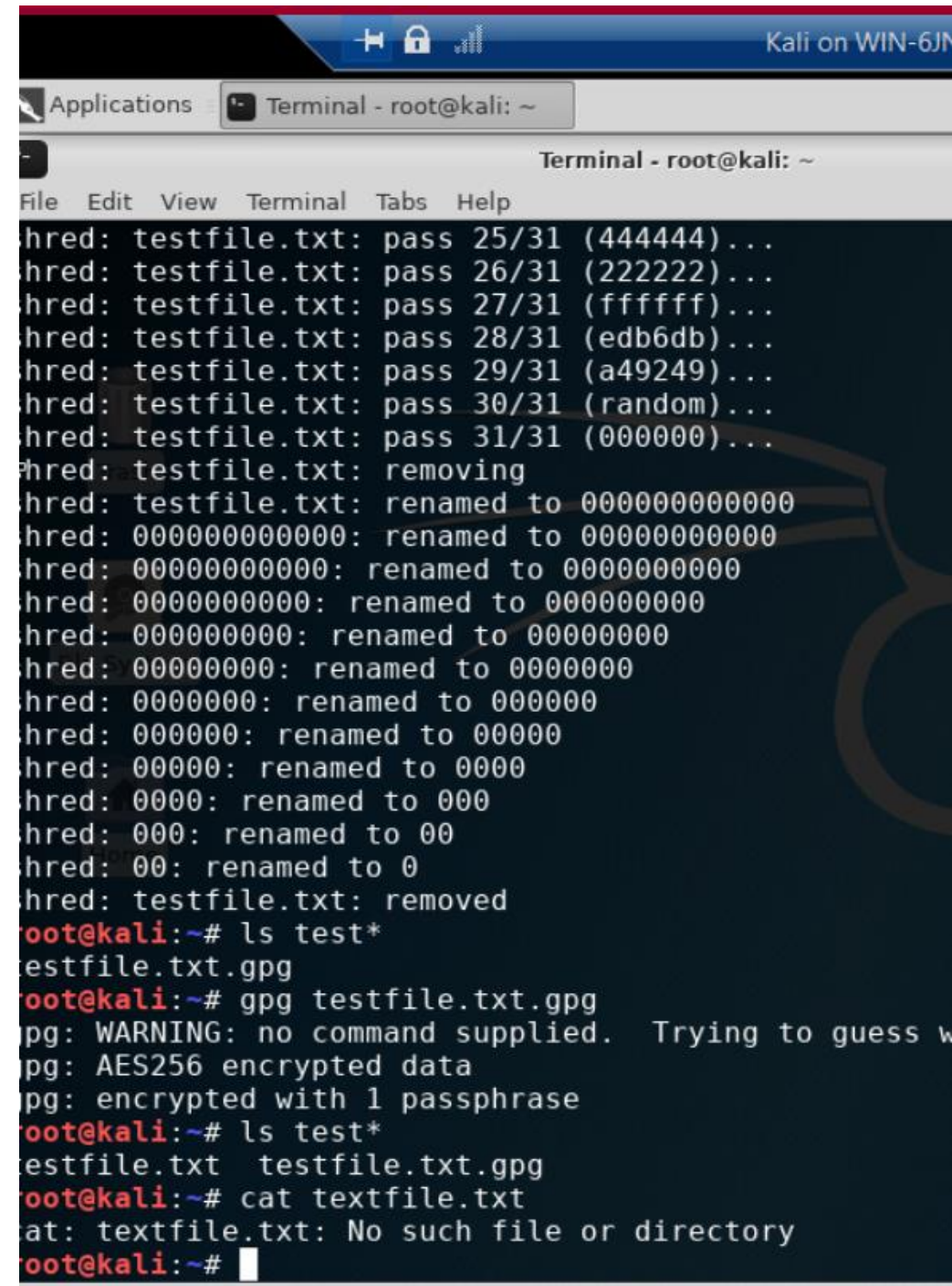
- This screenshot illustrates the process of decrypting an encrypted file using GPG and verifying the results.

## What's Shown:

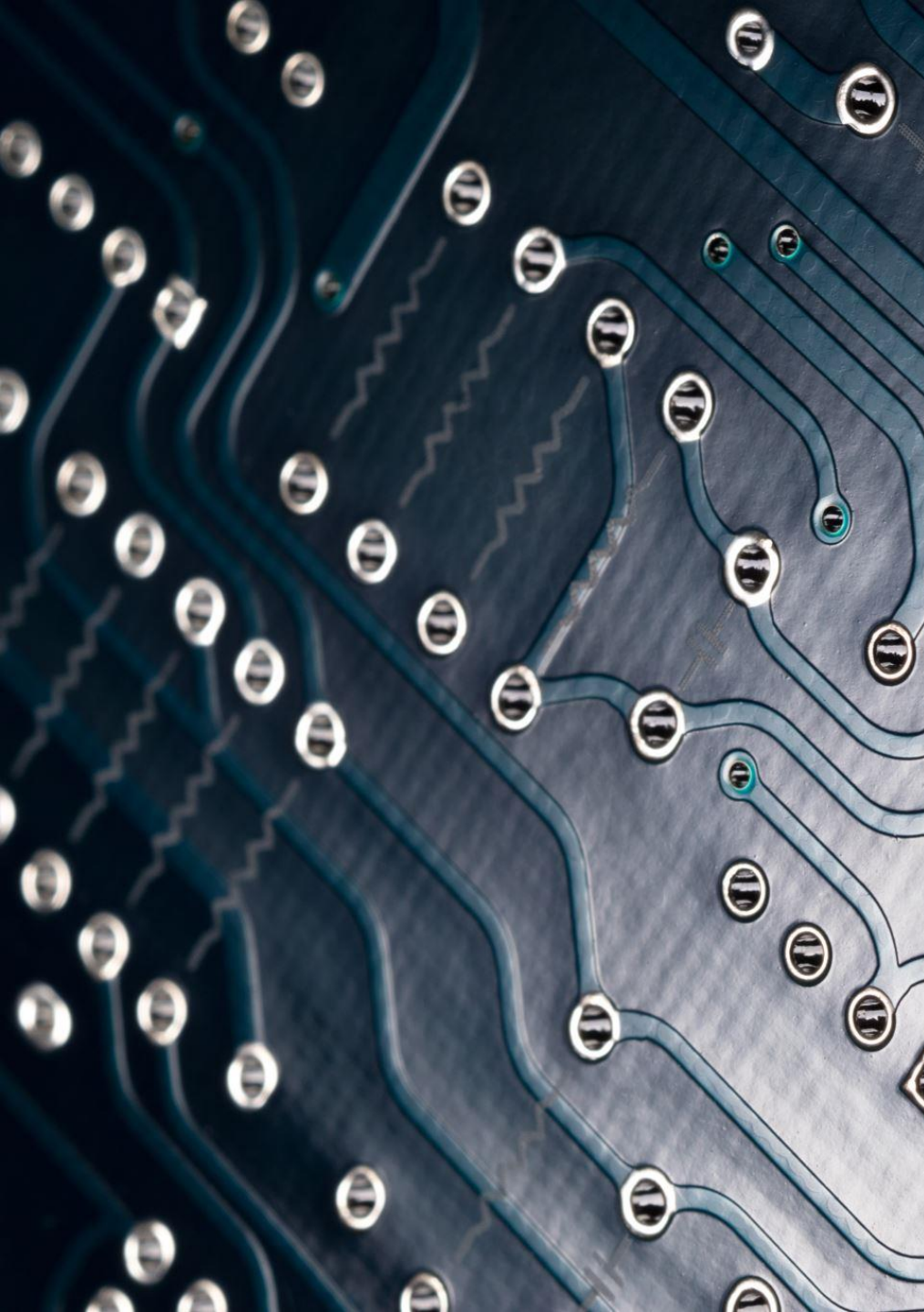
- Listing the Encrypted File:**
  - The command `ls test*` shows the encrypted file `testfile.txt.gpg`, confirming its existence.
- Decrypting the File:**
  - The command `gpg testfile.txt.gpg` was used to initiate the decryption process.
  - The decryption process was successfully started, using the AES256 encryption method with a passphrase.
- Listing Both Encrypted and Plaintext Files:**
  - After decryption, the `ls` command shows both the encrypted file (`testfile.txt.gpg`) and the restored plaintext file (`testfile.txt`).
  - This confirms that the file was successfully decrypted and the plaintext version has been recovered.

## Result:

- The decryption process was completed, with both the encrypted and plaintext files now listed together.



```
Kali on WIN-6JN
Applications Terminal - root@kali: ~
Terminal - root@kali: ~
File Edit View Terminal Tabs Help
hred: testfile.txt: pass 25/31 (444444)...
hred: testfile.txt: pass 26/31 (222222)...
hred: testfile.txt: pass 27/31 (ffffff)...
hred: testfile.txt: pass 28/31 (edb6db)...
hred: testfile.txt: pass 29/31 (a49249)...
hred: testfile.txt: pass 30/31 (random)...
hred: testfile.txt: pass 31/31 (000000)...
hred: testfile.txt: removing
hred: testfile.txt: renamed to 000000000000
hred: 000000000000: renamed to 000000000000
hred: 000000000000: renamed to 000000000000
hred: 000000000000: renamed to 0000000000
hred: 0000000000: renamed to 00000000
hred: 00000000: renamed to 0000000
hred: 0000000: renamed to 000000
hred: 000000: renamed to 00000
hred: 00000: renamed to 0000
hred: 0000: renamed to 000
hred: 000: renamed to 00
hred: 00: renamed to 0
hred: testfile.txt: removed
root@kali:~# ls test*
testfile.txt.gpg
root@kali:~# gpg testfile.txt.gpg
gpg: WARNING: no command supplied. Trying to guess w
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
root@kali:~# ls test*
testfile.txt testfile.txt.gpg
root@kali:~# cat testfile.txt
cat: testfile.txt: No such file or directory
root@kali:~#
```



# What I Learned

In Module 2, I successfully demonstrated how asymmetric encryption using GPG can be used to secure sensitive data. The encryption process ensured that the content of the file was protected from unauthorized access, while the shred command provided an additional layer of security by permanently removing the original plaintext file. This module emphasized the importance of securing data at rest through encryption and secure deletion practices.

While encrypting and protecting sensitive data is crucial, it's equally important to control and limit access to enterprise resources. In the next module, I will focus on enhancing system security by setting up a **stateful firewall**, which acts as a protective barrier to block unauthorized network traffic. This will further strengthen the overall security of the system by preventing potential attacks.





# Implementing a Stateful Firewall

In Module 3, I focused on improving network security by setting up a **stateful firewall** using iptables in a Linux environment. A stateful firewall tracks the state of active connections and allows or blocks traffic based on predefined security rules. This ensures that only trusted, established connections are allowed, while unauthorized traffic is blocked.

## Key Concepts:

- **TCP vs. UDP:**
  - TCP is a connection-oriented protocol that requires a three-way handshake to establish communication.
  - UDP is a connectionless protocol often used for quick data transfers, like DNS and DHCP.
- **Stateful Firewall:** Tracks the state of each network connection and allows only traffic that is part of an established, valid session.

# Effect of the INPUT DROP Policy in iptables

What effect does the `sudo iptables --policy INPUT DROP` command have on the What effect does the `sudo iptables --policy INPUT DROP` command have on access to computing resources?

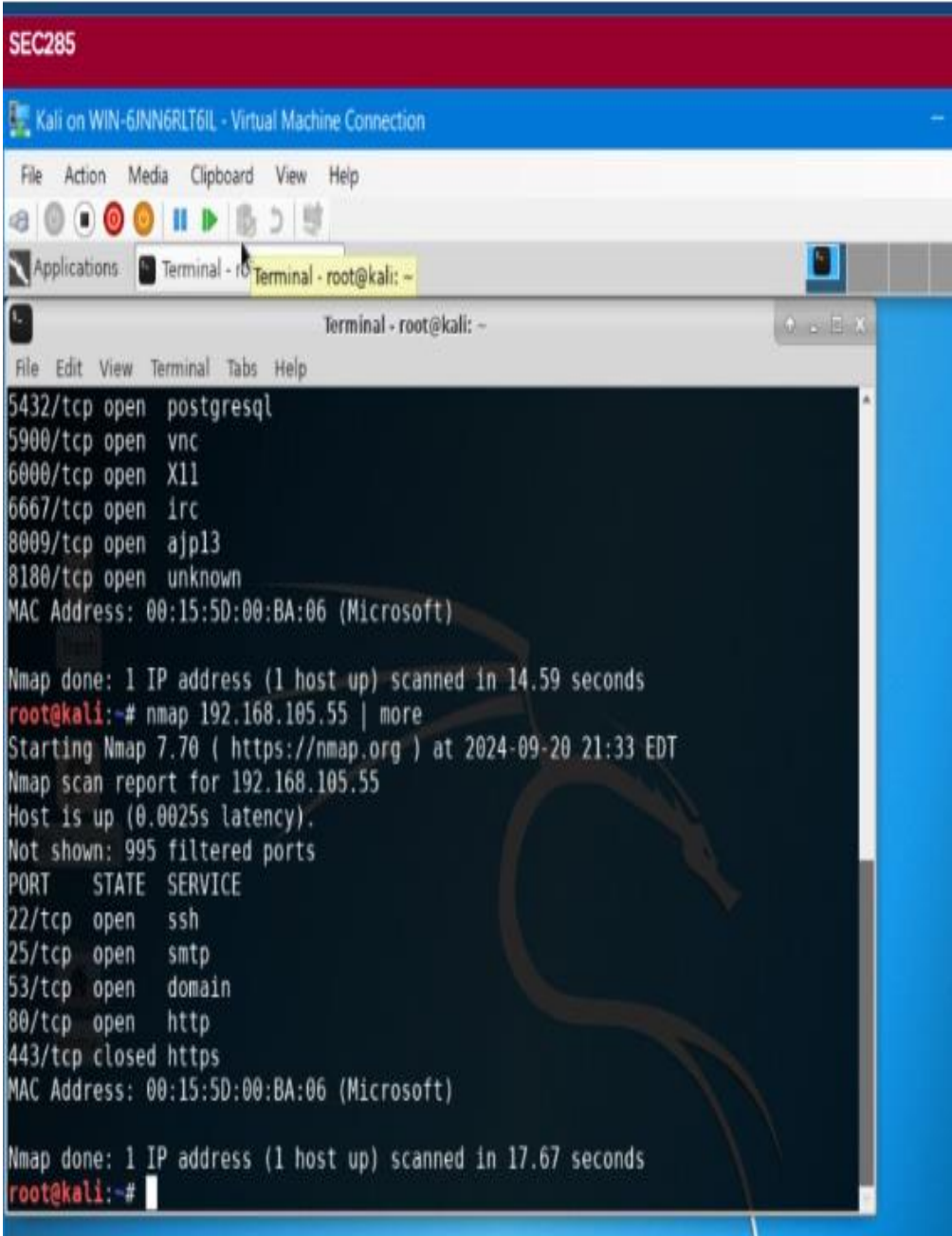
- The command `sudo iptables --policy INPUT DROP` blocks all incoming traffic to your computer by default. This means that any new connections, such as attempts to access your website or services, will be automatically denied unless specific rules are set to allow certain types of traffic. This is highly effective for enhancing security by preventing unwanted or malicious access, but it may also block important connections if the firewall rules aren't configured correctly

## •References:

(n.d.). The netfilter.Org "iptables" project. The Netfilter Webmasters. Retrieved September 20, 2024, from <https://www.iptables.org/projects/iptables/index.html>







```
SEC285
Kali on WIN-6JNN6RLT6IL - Virtual Machine Connection
File Action Media Clipboard View Help
Applications: Terminal - root@kali: -
Terminal - root@kali: -
File Edit View Terminal Tabs Help
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 00:15:5D:00:BA:06 (Microsoft)
Nmap done: 1 IP address (1 host up) scanned in 14.59 seconds
root@kali:~# nmap 192.168.105.55 | more
Starting Nmap 7.70 ( https://nmap.org ) at 2024-09-20 21:33 EDT
Nmap scan report for 192.168.105.55
Host is up (0.0025s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
443/tcp   closed https
MAC Address: 00:15:5D:00:BA:06 (Microsoft)
Nmap done: 1 IP address (1 host up) scanned in 17.67 seconds
root@kali:~#
```

# Nmap Scan Results of the Linux Server VM

Overview: The screenshot illustrates the results of an Nmap scan conducted on the Linux Server VM. Nmap is a powerful tool used to discover hosts and services on a network by scanning for open ports.

## Key Points:

- **Target IP Address:**
  - The scan was performed on the IP address 192.168.105.55.
- **Open Ports:**
  - The scan detected several open ports on the target, including:
    - Port 22 (SSH)
    - Port 25 (SMTP)
    - Port 53 (DNS)
    - Port 80 (HTTP)
    - Port 443 (HTTPS)
- **Filtered Ports:**
  - The scan also showed that 995 other ports were filtered, indicating that they were blocked or inaccessible.
- **Result:**
  - The Nmap scan successfully identified the services running on the open ports of the Linux Server VM, providing insight into its network configuration and potential vulnerabilities.



# What I Learned

In Module 3, I learned how to enhance network security by configuring a stateful firewall using **iptables**. I also gained experience in using the `sudo iptables --policy INPUT DROP` command to block all incoming traffic and then setting specific rules to allow or deny connections. Additionally, I used **Nmap** to scan the Linux Server VM, identifying open ports and potential vulnerabilities. This reinforced the importance of carefully managing network access to protect against unauthorized traffic.

Next, in Module 4, I will shift focus from firewall and network security to developing a **Bring Your Own Device (BYOD) Security Policy**. This policy is essential to regulate and secure personal devices accessing corporate networks, ensuring that security standards are maintained across all endpoints.





# Bring Your Own Device (BYOD) Security Policy

In Module 4, the focus shifts to creating a **Bring Your Own Device (BYOD) Security Policy**. This policy is designed to address the security risks posed by employees using personal devices (like smartphones, laptops, and tablets) to access corporate networks and data. BYOD policies ensure that appropriate security measures are in place to protect sensitive information while allowing flexibility for users.

## Key Concepts:

- BYOD Risks:** Personal devices may introduce malware, unauthorized access, or data leaks if not properly managed.
- Security Controls:** The policy includes requirements like device encryption, remote wiping, strong passwords, and security patches.
- SANS Policy Template:** A structured template used to create a comprehensive policy that addresses key areas like acceptable use, security requirements, and enforcement.

# BYOD Security Policy Overview & Purpose

## 1. Overview:

Many employees use their own devices like smartphones, tablets, and laptops at work. While this is convenient, it can introduce security risks such as viruses, unsecured connections, and accidental data sharing. To address these risks, ABC Corporation has created this Bring Your Own Device (BYOD) Security Policy.

## 2. Purpose:

The goal of this policy is to protect ABC Corporation's data and computer systems by managing how personal devices are used at work. By following this policy, we aim to prevent potential security problems that could arise from using personal devices.



# BYOD Security Policy Scope & Policy

## 3. Scope:

This policy applies to all employees, contractors, and third-party users who access ABC Corporation's computer systems using personal devices. This includes devices like smartphones, tablets, and laptops. The policy covers aspects like device registration, security updates, and data encryption.

## 4. Policy:

- **Device Registration:** All personal devices must be registered with the IT department before they can access corporate networks.
- **Security Updates:** Devices must have the latest security updates and antivirus software installed.
- **Authentication:** Multi-factor authentication (MFA) is required for all logins to corporate resources.
- **Data Access:** Access to sensitive data must be through secure, encrypted channels, such as a Virtual Private Network (VPN).
- **Data Storage:** Users are not allowed to store confidential corporate data on personal devices unless approved by IT.
- **Device Configuration:** Devices must be set to automatically lock after a period of inactivity.

# **BYOD Security Policy Compliance & Related Standards, Policies, and Processes**

## **5. Policy Compliance:**

The Information Security team will monitor compliance with this policy through regular audits and monitoring tools. Employees who violate this policy may face disciplinary actions, including losing access privileges or even termination of employment.

## **6. Related Standards, Policies, and Processes:**

This policy is linked to ABC Corporation's data privacy policy and other internal IT security policies. It also complies with external regulatory standards like HIPAA (Health Insurance Portability and Accountability Act) and PCI-DSS (Payment Card Industry Data Security Standard).



# BYOD Security Policy Definitions and Terms & Revision History

## 7. Definitions and Terms:

- **BYOD:** Bring Your Own Device, a policy that allows employees to use personal devices for work purposes.
- **CIA:** Confidentiality, Integrity, Availability - the core principles of information security.
- **Mobile Devices:** Personal devices such as smartphones, tablets, and laptops capable of accessing corporate resources.

## 8. Revision History:

Date of change	Responsible	Summary of change
August 2019	SANS policy team	Updated and converted to new format
September 2024	IT Security Team (Hector Acosta)	Updated



# What I Learned

In Module 4, I learned the importance of creating a comprehensive **BYOD (Bring Your Own Device) Security Policy** to safeguard an organization's network and data. This policy ensures that personal devices, like smartphones and laptops, used by employees follow strict security protocols such as device registration, regular security updates, multi-factor authentication, and data encryption. I also learned how security policies help mitigate risks associated with personal devices accessing corporate resources, while still allowing employees flexibility in their work environment.

In the next module, I will dive deeper into user authentication and protection by implementing **Multi-factor Authentication (MFA)**. MFA adds an extra layer of security to ensure that only authorized users can access corporate resources, further reducing the risks of unauthorized access or breaches.



# Implementing Multi-Factor Authentication (MFA)

In Module 5, the focus shifts to strengthening user authentication by implementing **Multi-Factor Authentication (MFA)**. MFA is a security enhancement that requires users to provide two or more verification factors to gain access to corporate resources. By using MFA, organizations can significantly reduce the risk of unauthorized access, even if a user's password is compromised.

## Key Concepts:

- **Multi-Factor Authentication (MFA):** Requires a combination of factors such as something you know (password), something you have (one-time password or token), and something you are (biometrics).
- **Time-Based One-Time Password (TOTP):** One method of MFA, which generates a one-time code that is valid only for a short period.
- **Enhanced Security:** MFA adds an additional layer of defense beyond passwords, making it much harder for attackers to compromise user accounts.



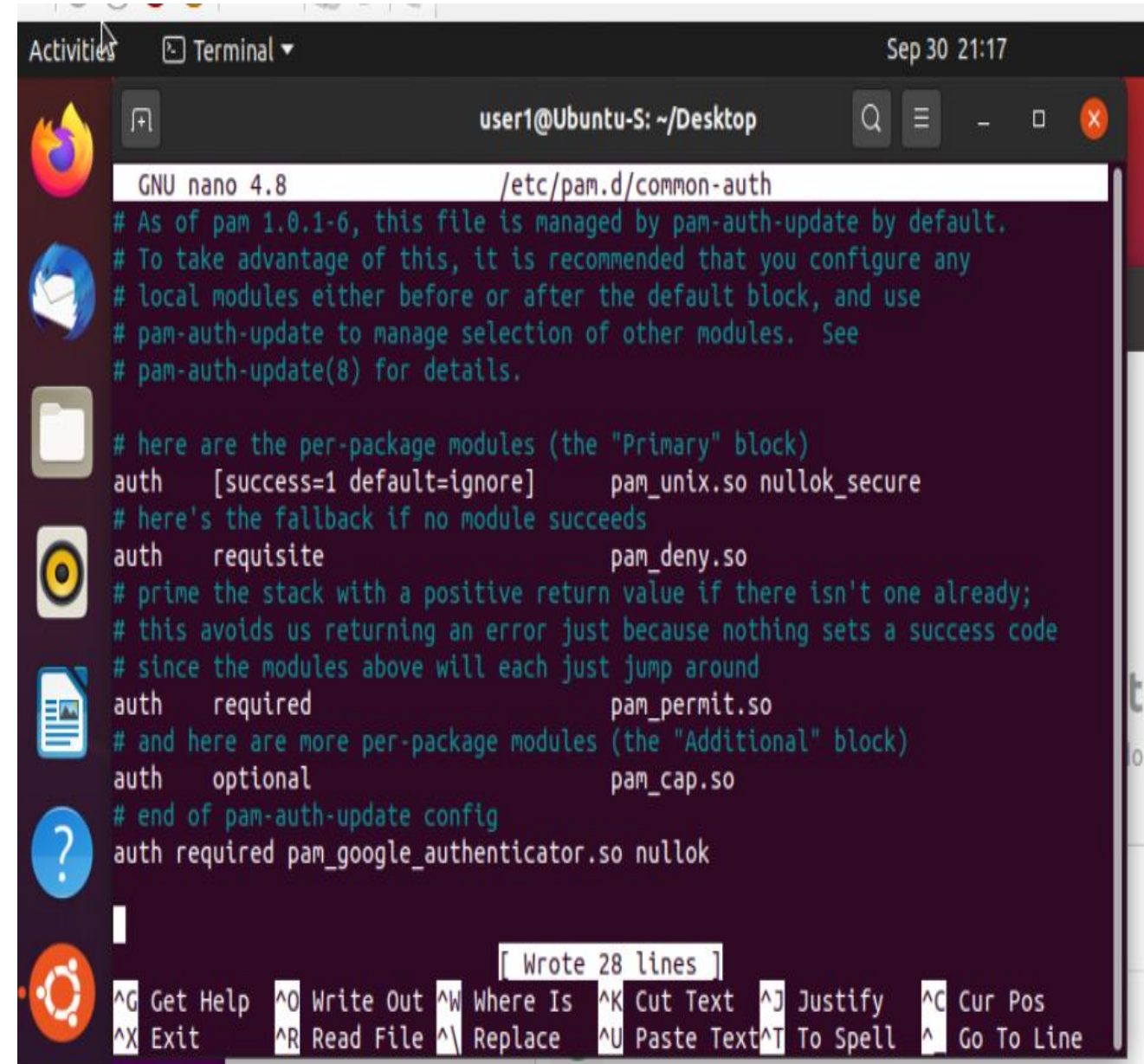


# Common-auth Configuration File

**Overview:** This screenshot shows the configuration of the `/etc/pam.d/common-auth` file, where the **Google Authenticator** module was added to enable Multi-Factor Authentication (MFA) on the Ubuntu system.

**Key Step:**

- The line `auth required pam_google_authenticator.so nullok` was added to the bottom of the file. This ensures that the Google Authenticator OTP will be required for user authentication, adding an extra layer of security beyond the standard password.



```
GNU nano 4.8 /etc/pam.d/common-auth
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
auth [success=1 default=ignore] pam_unix.so nullok_secure
# here's the fallback if no module succeeds
auth requisite pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth required pam_permit.so
# and here are more per-package modules (the "Additional" block)
auth optional pam_cap.so
# end of pam-auth-update config
auth required pam_google_authenticator.so nullok

[ Wrote 28 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

# MFA Logon Screen

**Overview:** This screenshot displays the logon screen where the **Multi-Factor Authentication (MFA)** process is in place. After configuring the Google Authenticator module, the system now requires both the regular password and a time-based one-time password (OTP) generated by the Google Authenticator app to log in.

**Key Step:**

- When logging in, the user must first enter their usual password, followed by the verification code (OTP) provided by the Google Authenticator app. This ensures that only users who have both credentials can access the system.





# What I Learned

In Module 5, I learned how to implement **Multi-Factor Authentication (MFA)** using the **Google Authenticator** app to add an extra layer of security. This module taught me how to:

- Configure Ubuntu to require a time-based one-time password (OTP) alongside a regular password during the login process.
- Set up Google Authenticator on a mobile device and integrate it with the Ubuntu system.
- Secure the system with MFA, making it harder for unauthorized users to gain access, even if the password is compromised.

In the next module, I will focus on **Vulnerability Assessment**, where I will use tools like Nmap, Netcat, and Wireshark to identify security vulnerabilities in the system. These tools will help analyze the security posture of the system and pinpoint areas that require additional attention.



# Vulnerability Assessment



In Module 6, I focused on conducting a comprehensive **Vulnerability Assessment** to evaluate the security posture of the Linux Server VM. This was done using various industry-standard tools like **Nmap**, **Ncat**, **Wireshark**, and **Nessus Essentials**. These tools help identify security vulnerabilities and provide insight into areas that need attention to improve overall system security.

## Key Steps:

- **IP Address Identification:** Used the `ifconfig` command to identify the IP addresses of the Linux Server and Kali VMs.
- **Nmap Scanning:** Performed a network scan using **Nmap** to identify hosts and services running on the network.
- **Banner Grabbing with Ncat:** Gathered information about FTP, SSH, and SMTP services using **Ncat**.
- **Packet Sniffing with Wireshark:** Analyzed network traffic to capture Telnet credentials, demonstrating how unencrypted data can be intercepted.
- **Vulnerability Scanning with Nessus:** Conducted a basic network vulnerability scan using **Nessus Essentials** to assess the Linux Server VM's security.

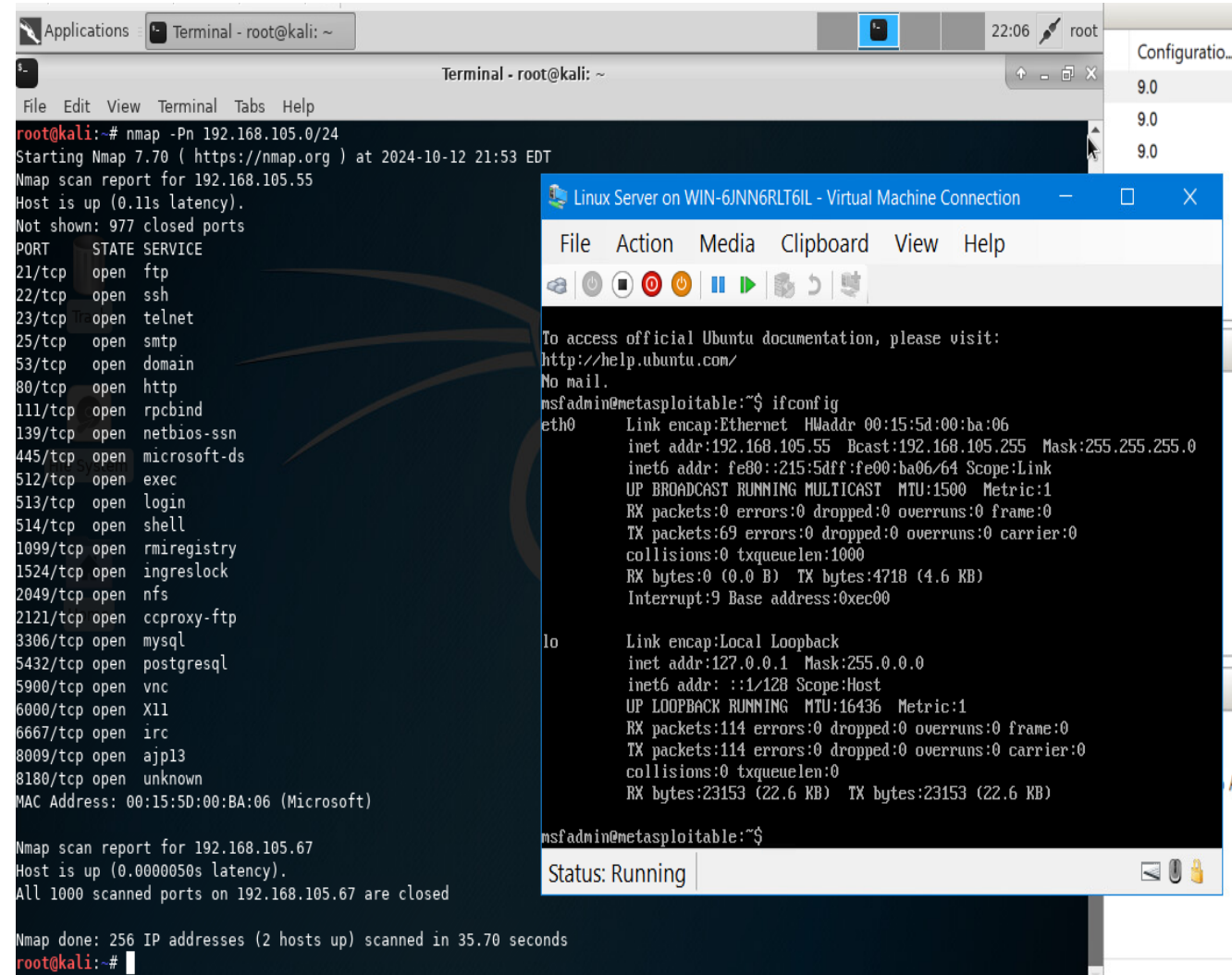
# Nmap Scan Results

**Overview:** The screenshot shows the results of an **Nmap** scan conducted on the network that includes both the **Kali VM** and the **Linux Server VM**. Nmap is a tool used to discover hosts and services on a network by sending packets and analyzing responses.

## Key Details:

- The scan identified both the **Kali VM** (IP: 192.168.105.55) and the **Linux Server VM** (IP: 192.168.105.67).
- The scan shows several open ports on both VMs:
  - **Linux Server VM:** Ports 21 (FTP), 22 (SSH), 23 (Telnet), 25 (SMTP), and others.
  - **Kali VM:** Ports 80 (HTTP) and 443 (HTTPS), among others.
- Nmap also indicated that 977 ports were closed on the Linux Server VM, demonstrating a secure configuration.

**Result:** The Nmap scan provides valuable insights into the services running on both VMs and highlights potential attack vectors that should be secured.



The screenshot displays a Kali Linux terminal window with the following Nmap scan results:

```
root@kali:~# nmap -Pn 192.168.105.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2024-10-12 21:53 EDT
Nmap scan report for 192.168.105.55
Host is up (0.11s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:15:5D:00:BA:06 (Microsoft)

Nmap scan report for 192.168.105.67
Host is up (0.0000050s latency).
All 1000 scanned ports on 192.168.105.67 are closed

Nmap done: 256 IP addresses (2 hosts up) scanned in 35.70 seconds
root@kali:~#
```

Overlaid on the terminal is a "Linux Server on WIN-6JNN6RLT6IL - Virtual Machine Connection" window. It shows network configuration details for the Linux Server VM:

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:ba:06
          inet addr:192.168.105.55  Bcast:192.168.105.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe00:ba06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:4718 (4.6 KB)
          Interrupt:9 Base address:0xec00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23153 (22.6 KB)  TX bytes:23153 (22.6 KB)

msfadmin@metasploitable:~$
```

The status bar at the bottom of the VM connection window indicates "Status: Running".

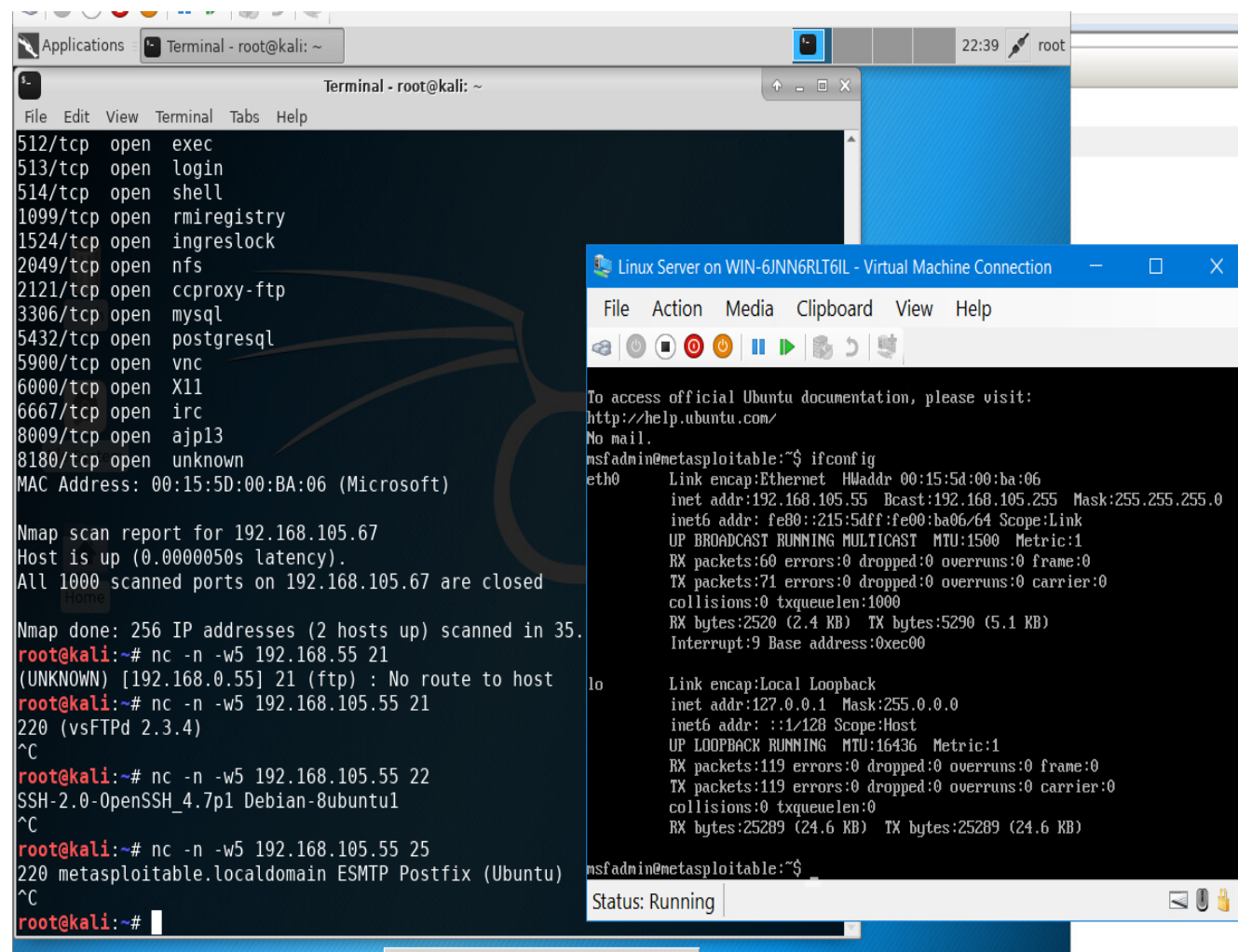
# Netcat Banner Grabbing

**Overview:** The screenshot shows the use of **Netcat** for banner grabbing on the **Linux Server VM** from the **Kali VM**. Banner grabbing is used to collect information about services running on a server, including version numbers, which can help in identifying potential vulnerabilities.

## Key Details:

- **FTP Banner:** The command `nc -n -w5 192.168.105.55 21` was used to grab the banner of the FTP service running on the Linux Server VM, identifying the FTP server.
- **SSH Banner:** The command `nc -n -w5 192.168.105.55 22` was used to grab the SSH service banner, showing the version of OpenSSH running on the Linux Server VM.
- **SMTP Banner:** The command `nc -n -w5 192.168.105.55 25` was used to grab the SMTP service banner, identifying the Postfix mail server on the Linux Server VM.

**Result:** By using Netcat, I was able to successfully capture the version information of FTP, SSH, and SMTP services running on the Linux Server VM. This information is valuable in identifying potential vulnerabilities based on outdated or insecure service versions.



The screenshot displays two overlapping windows. The background window is a terminal titled 'Terminal - root@kali: ~' showing the output of an Nmap scan and subsequent netcat connections. The foreground window is titled 'Linux Server on WIN-6JNN6RLT6IL - Virtual Machine Connection' and shows the output of the 'ifconfig' command on the Linux Server VM.

```
File Edit View Terminal Tabs Help
512/tcp open  exec
513/tcp open  login
514/tcp open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 00:15:5D:00:BA:06 (Microsoft)

Nmap scan report for 192.168.105.67
Host is up (0.000050s latency).
All 1000 scanned ports on 192.168.105.67 are closed

Nmap done: 256 IP addresses (2 hosts up) scanned in 35.
root@kali:~# nc -n -w5 192.168.55 21
(UNKNOWN) [192.168.0.55] 21 (ftp) : No route to host
root@kali:~# nc -n -w5 192.168.105.55 21
220 (vsFTPd 2.3.4)
^C
root@kali:~# nc -n -w5 192.168.105.55 22
SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
^C
root@kali:~# nc -n -w5 192.168.105.55 25
220 metasploitable.localdomain ESMTPostfix (Ubuntu)
^C
root@kali:~#
```

The foreground window shows the output of the 'ifconfig' command:

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:ba:06
          inet addr:192.168.105.55  Bcast:192.168.105.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe00:ba06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2520 (2.4 KB)  TX bytes:5290 (5.1 KB)
          Interrupt:9 Base address:0xec00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:119 errors:0 dropped:0 overruns:0 frame:0
          TX packets:119 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25289 (24.6 KB)  TX bytes:25289 (24.6 KB)

msfadmin@metasploitable:~$
```

The status bar at the bottom of the foreground window indicates 'Status: Running'.



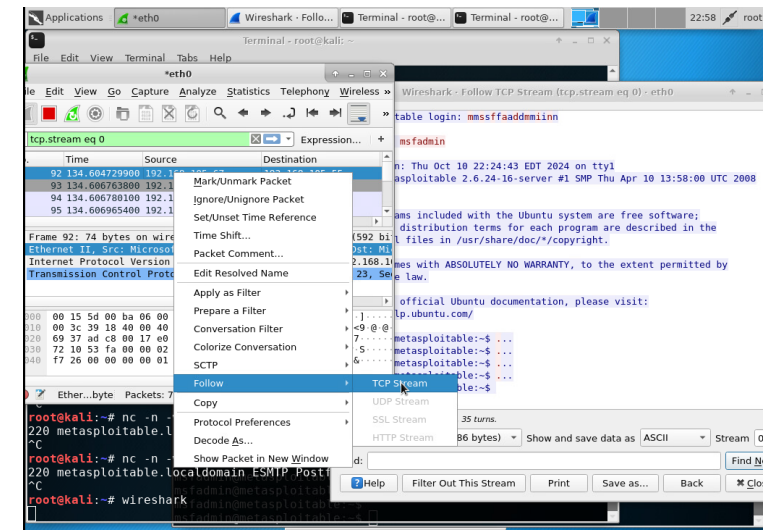
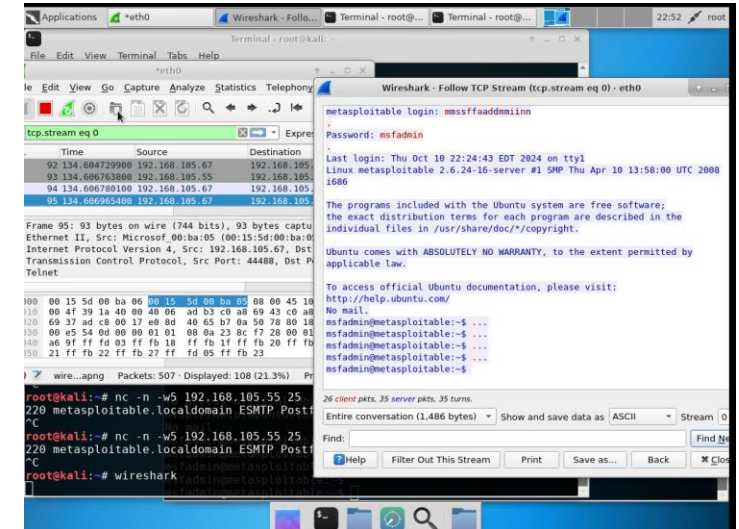
# Wireshark - Capturing Telnet Credentials

**Overview:** The screenshots demonstrate how **Wireshark** was used to capture network traffic between the **Kali VM** and the **Linux Server VM**. By analyzing the captured Telnet packets, we were able to observe the plaintext transmission of sensitive information such as the **Telnet username and password**.

## Key Details:

- **Packet Capture:**
  - The Telnet communication between the Kali VM and the Linux Server VM was captured using Wireshark.
  - The traffic was filtered by selecting the **telnet** packets for analysis.
- **Follow TCP Stream:**
  - Using the "Follow TCP Stream" option, Wireshark revealed the full Telnet session, including the **username** (msfadmin) and **password** (msffadddmiinn) transmitted in plaintext.
  - This demonstrates the vulnerability of using Telnet, which sends login credentials over the network without encryption, making them easily accessible to anyone monitoring the traffic.

**Result:** The screenshots clearly show the Wireshark **Follow TCP Stream** feature displaying the plaintext credentials, highlighting the security risk of using unencrypted communication protocols like Telnet.



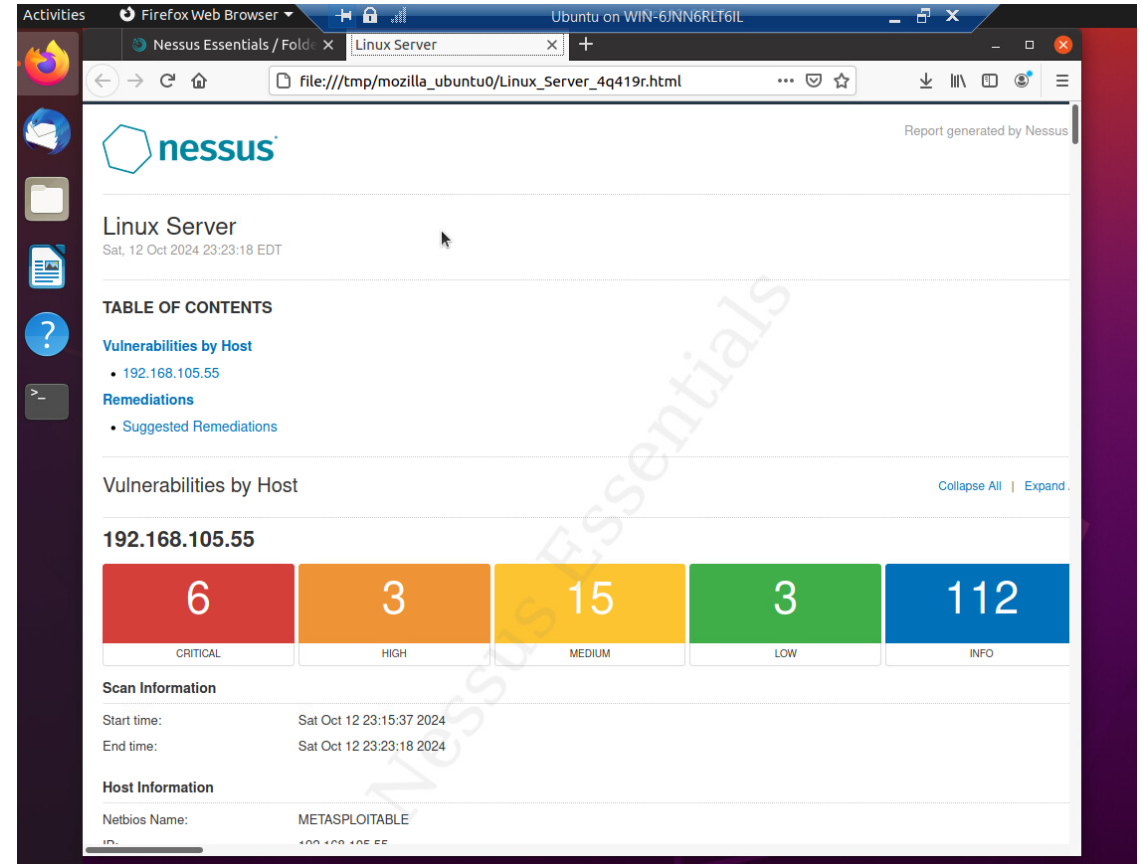
# Nessus Vulnerability Scan Report

**Overview:** The screenshot shows the high-level view of the **Nessus vulnerability scan report** for the **Linux Server VM**. Nessus Essentials is a powerful tool used for scanning systems to identify vulnerabilities and provide a comprehensive security assessment.

## Key Details:

- **Vulnerabilities by Host:** The Nessus scan of the Linux Server (IP: 192.168.105.55) identified several vulnerabilities, categorized by their severity:
  - **6 Critical**
  - **3 High**
  - **15 Medium**
  - **3 Low**
  - **112 Informational**
- The critical and high vulnerabilities need immediate attention to secure the system, while medium and low issues can be addressed subsequently.

**Result:** The scan highlights the critical areas where the system is vulnerable and provides recommendations for remediation. This information is vital for prioritizing security fixes to protect the Linux Server VM from potential threats.



# Challenges Faced

## Technical Challenges:

- **Firewall Configuration:** Setting up and fine-tuning the **stateful firewall** with iptables required careful configuration to ensure only authorized traffic was allowed while maintaining the system's functionality.
- **Multi-Factor Authentication (MFA) Setup:** Configuring Google Authenticator and linking it with the Ubuntu system for MFA involved navigating various steps, including correctly modifying authentication files.
- **Vulnerability Scanning:** Using multiple tools like **Nmap**, **Ncat**, and **Nessus** required interpreting detailed scan results, which sometimes produced overwhelming data to analyze.

## Time Management:

- Balancing the different modules and ensuring that each step was completed within the required timeframe posed a significant challenge, particularly when troubleshooting technical issues.

## Tool Familiarity:

- Gaining familiarity with tools like **Wireshark** and **Nessus** took time, especially in learning how to interpret the detailed packet captures and vulnerability reports effectively.

## Security Risks:

- Identifying and understanding the risks posed by unsecured protocols like **Telnet** and learning how to mitigate these risks through encryption and secure authentication practices was a critical learning curve.



# Career Skills Gained

## 1. Network Security and Firewall Management:

- Gained experience in setting up and managing **stateful firewalls** using iptables to control traffic flow and secure network connections.
- Learned to configure firewalls to allow only authorized access, enhancing overall system security.

## 2. Multi-Factor Authentication (MFA):

- Implemented **MFA** using Google Authenticator to provide an additional layer of security beyond passwords.
- Acquired skills in configuring authentication protocols and integrating mobile-based OTP systems with Linux servers.

## 3. Vulnerability Assessment:

- Developed proficiency in using tools like **Nmap**, **Netcat**, **Wireshark**, and **Nessus** to scan systems, identify vulnerabilities, and assess the overall security posture of a network.
- Learned how to interpret scan results and prioritize remediation efforts based on vulnerability severity.

## 4. Incident Response and Packet Analysis:

- Gained hands-on experience in capturing and analyzing network traffic using **Wireshark** to identify potential security threats, including the use of insecure protocols like Telnet.
- Enhanced skills in following network activity and tracing security vulnerabilities through packet analysis.

## 5. Secure Configuration and System Hardening:

- Learned to apply security best practices to harden systems, such as disabling insecure services, configuring secure authentication, and ensuring system configurations are up to date.

# Career Skills Gained

## Soft Skills:

### Problem-Solving:

- Developed strong problem-solving skills by troubleshooting network and security issues, such as resolving firewall configuration errors and interpreting complex vulnerability reports.

### Time Management:

- Managed multiple tasks across different modules and tools, ensuring that each project milestone was completed within the required timeframe.

### Attention to Detail:

- Gained a deep understanding of the importance of precision when working with security configurations, analyzing network traffic, and interpreting vulnerability data to prevent security breaches.

### Adaptability:

- Adapted quickly to new tools and processes, learning to efficiently use **Nmap**, **Netcat**, **Wireshark**, and **Nessus** for the first time while troubleshooting issues as they arose.

# Conclusion

Throughout this project, I gained valuable experience in various cybersecurity practices essential for protecting and securing enterprise systems. From setting up **firewalls** and implementing **Multi-Factor Authentication (MFA)** to conducting detailed **vulnerability assessments** with tools like **Nmap**, **Netcat**, **Wireshark**, and **Nessus**, I developed both technical and analytical skills crucial for identifying and mitigating security threats.

## Key Takeaways:

- **Network Security:** Understanding how to secure network traffic and prevent unauthorized access using firewalls and vulnerability scanning tools.
- **Authentication:** Learning the importance of secure authentication methods through MFA and its role in reducing risks associated with single-factor authentication.
- **Vulnerability Management:** Gaining proficiency in identifying, analyzing, and addressing system vulnerabilities to strengthen the overall security posture.

This project has prepared me to apply these skills in real-world environments, ensuring that systems are secure, risks are mitigated, and potential threats are effectively managed. Moving forward, I am eager to continue enhancing my cybersecurity skills and contributing to the protection of critical systems and data.



# References

## **ACI Learning | Infosec Learning Inc.**

- Course materials and labs provided throughout each module for hands-on practice and guidance.

## **Module Videos:**

- Instructional videos provided within each module to guide through the setup, configuration, and execution of cybersecurity tasks, including firewall setup, MFA configuration, vulnerability scanning, and packet analysis.