SSH is commonly used for remote access to Linux servers. Securing it is critical to prevent unauthorized access.

Disable Root Login via SSH

sudo nano /etc/ssh/sshd_config

Change the Following Line:

PermitRootLogin no

Change the SSH Port (For Obscurity)

sudo nano /etc/ssh/

sshd_config
Change the Following Line:
Port 2222

# Allow Only Specific Users to SSH In
sudo nano /etc/ssh/sshd_config
# Add the Following Line (Replace `username` with the Allowed User)
AllowUsers username

Restart the SSH Service to

Apply Changes
sudo systemctl restart sshd

Use SSH Keys Instead of Passwords for Authentication
Generate an SSH Key Pair on Your Local Machine
ssh-keygen -t rsa -b 4096

Copy the Public Key to the Remote Server
ssh-copy-id username@remote-server

# Disable Password Authentication
sudo nano /etc/ssh/sshd_config
# Change the Following Line:
PasswordAuthentication no

# Restart the SSH Service Again
sudo systemctl restart sshd
Tip: Regularly monitor the SSH logs (/var/log/auth.log) for any suspicious login

attempts.

4. Configuring the Firewall with ufw

The **Uncomplicated Firewall (UFW)** is a user-friendly interface for managing firewall rules. It helps control which services can access your Linux system.

# Enable the Firewall (After Configuring Rules)
sudo ufw enable

# Check the Current Status of the Firewall
sudo ufw status verbose

# Allow Incoming Traffic on a Specific Port (e.g., HTTP on Port 80)
sudo ufw allow 80/tcp

# Allow SSH on a Non-Standard Port (e.g., 2222)
sudo ufw allow 2222/tcp

Deny Incoming Traffic on a

Specific Port (e.g., FTP on Port 21)
sudo ufw deny 21/tcp

# Allow a Specific IP Address to Access a Port
sudo ufw allow from 192.168.1.100 to any port 22

# Delete a Specific Firewall Rule
sudo ufw delete allow 80/tcp

Disable the Firewall (Use

with Caution)
sudo ufw disable
Tip: Always test your firewall rules carefully to ensure you don’t lock yourself out of the system.

5. Regular System Updates and Patching

Keeping your system and software up-to-date is essential for security.

Regular updates help patch vulnerabilities and fix bugs.
# Update the Package List
sudo apt update

# Upgrade All Installed Packages
sudo apt upgrade

# Perform a Full Distribution Upgrade (Includes Kernel Updates)
sudo apt full-upgrade

# Remove Unnecessary Packages
sudo apt autoremove

# Install Security Updates Automatically (Optional)
sudo apt install unattended-upgrades
sudo dpkg-reconfigure --priority=low unattended-upgrades
***Tip**: Enable automatic security updates on critical servers to ensure they're*

always protected against known vulnerabilities.

6. Monitoring and Logging System Activity

Regularly monitoring your system can help detect unusual activity and potential security breaches.

```
# View System Logs
sudo less /var/log/syslog
```

# View Authentication Logs (e.g., SSH Login Attempts)
sudo less /var/log/auth.log

# Check Disk Usage (To Monitor for Suspicious File Activity)
sudo du -sh /path/to/directory

# Monitor Active Network Connections
sudo netstat -tuln

Monitor Running Processes in Real-Time
top

# Display the Last 100 Login Attempts
sudo last -n 100

# Set Up Log Rotation to Manage Log File Size
sudo nano /etc/logrotate.conf
*Tip: Consider using tools like **Fail2ban** to monitor logs*

and automatically block IPs with too many failed login attempts.

7. Securing Network Services

Every network service running on your Linux system is a potential entry point for attackers. Here’s how to secure them:

Disable Unnecessary Services

Check the Status of a

Service
sudo systemctl status
servicename

# Stop a Service
sudo systemctl stop
servicename

Disable a Service from
Starting at Boot
sudo systemctl disable
servicename

Enable a Service (When

Needed)
sudo systemctl enable servicename

# Restrict Access to Specific Services Using TCP Wrappers
# Edit the /etc/hosts.allow File
sudo nano /etc/hosts.allow
# Example: Allow SSH from a Specific IP
sshd: 192.168.1.100

# Edit the /etc/hosts.deny File
sudo nano /etc/hosts.deny
# Example: Deny SSH from All Other IPs
sshd: ALL

# Use AppArmor or SELinux for Additional Security (Optional)
# Enable AppArmor (If Supported)
sudo systemctl enable apparmor

sudo systemctl start apparmor

Tip: Regularly audit running services and disable any that are not needed to reduce your system's attack surface.

8. Using fail2ban to Protect Against Brute-Force Attacks

Fail2ban monitors log files and bans IPs that show signs of malicious behavior, such as too many failed login attempts.

```
# Install Fail2ban
sudo apt install fail2ban

# Start the Fail2ban Service
sudo systemctl start fail2ban
sudo systemctl enable fail2ban

# Copy the Default
```

Configuration File to Create a Local Configuration
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

# Edit the Local Configuration File
sudo nano /etc/fail2ban/jail.local
# Configure Settings Like Ban Time, Max Retries, etc.

Restart the Fail2ban

Service to Apply Changes
sudo systemctl restart fail2ban

# Check the Status of Fail2ban and Active Jails
sudo fail2ban-client status
*Tip: Customize the configuration of **Fail2ban** to suit your needs and protect against various types of attacks.*

9. Limiting

Resource Usage with ulimit

ulimit can be used to limit the resources available to processes running on your system, helping to prevent abuse or accidental system crashes.

View the Current Limits for a Session

ulimit -a

Set a Limit for the

Maximum Number of Open Files
ulimit -n 1024

Set a Limit for the Maximum CPU Time (in Seconds) a Process Can Use
ulimit -t 600

Set a Limit for the Maximum Memory Usage
ulimit -m 1048576 # In KB

Set Limits Permanently by

Editing /etc/security/limits.conf

sudo nano /etc/security/limits.conf

Example: Limit the Number of Open Files for All Users

* hard nofile 1024

***Tip**: Use ulimit carefully to avoid unintended system behavior. Setting limits too low may prevent legitimate processes from functioning properly.*

10. Securing Web Servers and Applications

If you’re running web servers or applications on your Linux system, additional security measures are required.

Enable HTTPs with Let's Encrypt (Using Certbot)

sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d

yourdomain.com

# Secure Apache Web Server
# Disable Directory Listing
sudo nano /etc/apache2/apache2.conf
# Add the Following Line:
Options -Indexes

# Disable Unused Apache Modules
sudo a2dismod module_name

```
# Restart Apache
sudo systemctl restart
apache2

# Secure Nginx Web Server
# Disable Unnecessary
HTTP Methods
sudo nano /etc/nginx/
nginx.conf
# Add the Following:
if ($request_method !~
^(GET|POST)$) {
return 444;
}
```

```
# Restart Nginx
sudo systemctl restart nginx
```

Tip: Use security headers and configure your web server to minimize information disclosure and harden its configuration against common attacks.

Conclusion for Part 9

In **Part 9**, we've covered essential Linux security

practices, including user and file permissions management, securing SSH, configuring firewalls, monitoring system activity, and protecting network services. Implementing these practices will help you secure your Linux system and protect it from potential threats.

In **Part 10: Troubleshooting Common Linux Issues**, we'll focus on diagnosing and

resolving common Linux problems, such as boot issues, performance problems, and network connectivity issues.
Stay tuned!
A YouTube Channel for Cybersecurity Lab’s Poc and Write-ups

Cyberw1ng

Learn Cyber Security and Create Awareness ~

cyberwing Stay tuned with me, Subscribe, and Like the Videos... Ask Doubts...

www.youtube.com

Github for Resources:

Cyberw1ng — Overview

Security Researcher and Bug Hunter. Cyberw1ng has 8

repositories available. Follow their code on GitHub.

github.com

Telegram Channel for Free Ethical Hacking Dumps

Ethical Hacking Dumps — CEH, OSCP, Comptia Materials and Books for Ethical Hacking

Exams like CEH v12, OSCP, Comptia Pentest+, Comptia Security+, Comptia Network+...

t.me

Thank you for Reading!

Happy Ethical Hacking ~

Author: Karthikeyan Nagaraj

~ Cyberw1ng

1K