Operating Systems Lab Journal

Week 7 - Security Audit and System Evaluation

This journal records my Week 7 security audit activities. It covers Lynis security scanning, nmap network security assessment, access control verification, service audit (justifying all running services), and a system configuration review. Screenshots and command outputs are captured as evidence.

1. Objectives

The goal of Week 7 was to evaluate the overall security posture of my Linux server environment. To do this, I performed a structured security audit using standard tools and then reviewed system configuration items that are commonly associated with hardening, least privilege, and reduced attack surface.

Mandatory tasks completed this week:

- Security scanning with Lynis
- Network security assessment with nmap
- Access control verification (users, sudo privileges, SSH policies, password policies)
- Service audit (justify all running services)
- System configuration review (firewall, automatic updates, mandatory access control, logging)

2. Lab Environment and Scope

I used a two-VM VirtualBox lab. The Ubuntu Server VM was the primary audit target. The Ubuntu Workstation VM was used to run network scans against the server over the Host-Only network.

System	Role	OS / Version	Network adapters	IP address (Host-Only)
Server VM	Audit target	Ubuntu Server LTS	Adapter 1: NAT; Adapter 2: Host-Only	[e.g., 192.168.56.X]
Workstation VM	Scanner/Client	Ubuntu Desktop	Adapter 1: NAT; Adapter 2: Host-Only	[e.g., 192.168.56.Y]

Scope notes:

- The audit focuses on the server VM configuration, exposed network services, authentication controls, and hardening settings.
- Any changes made during the audit were re-checked (e.g., re-running service/port lists) to confirm the expected effect.
- Evidence was captured as terminal screenshots and saved command outputs in a local folder (e.g., ~/week7_audit).

3. Methodology and Evidence Collection

I followed a repeatable workflow: identify baseline state, scan using tools, review findings, verify configuration items manually, and then justify or remediate where appropriate. I used terminal output logging (tee) so results are saved and can be referenced in the report.

Evidence folder setup (run on both VMs):

```
mkdir -p ~/week7_audit
cd ~/week7_audit
```

IP discovery (server and workstation):

```
ip addr
ip -br addr | tee ~/week7_audit/<system>_ip.txt
```

4. Security Scanning with Lynis (Server VM)

Lynis provides a security audit that checks common hardening items such as SSH configuration, file permissions, service exposure, kernel settings, and installed security tooling. I installed Lynis from Ubuntu repositories and ran a full system audit.

Installation and version check:

```
sudo apt update && sudo apt -y upgrade
sudo apt install -y lynis
lynis --version | tee ~/week7_audit/lynis_version.txt
```

Audit execution and output capture:

```
sudo lynis audit system | tee ~/week7_audit/lynis_audit.txt
sudo cp /var/log/lynis.log ~/week7_audit/
sudo cp /var/log/lynis-report.dat ~/week7_audit/
sudo chown -R $USER:$USER ~/week7_audit
```

After running Lynis, I recorded the final summary (hardening index, warnings, and suggestions) as screenshots. I also extracted key lines from the report file to make it easier to discuss the most important issues in my journal.

Key finding extraction:

```
grep -E "hardening|warning|suggestion" -i /var/log/lynis-report.dat | head -n 80 | tee
~/week7_audit/lynis_key_findings.txt
```

Lynis results summary (fill using your screenshot/output):

Item	Observed result	Notes / actions taken
Hardening index	[__/100]	Recorded from Lynis summary screen; used to evaluate baseline security posture.
Total warnings	[__ warnings]	Reviewed each warning and matched it to configuration or packages.
Total suggestions	[__ suggestions]	Prioritised suggestions with

		highest security impact.
Top 3 warnings	[List 3 warning IDs/topics]	Explain what they mean and what I changed (if anything).
Top 3 suggestions	[List 3 suggestion IDs/topics]	Explain improvement opportunities and planned actions.

Evaluation: Lynis helped me confirm whether my server matches common security baselines. I used its warnings/suggestions as a checklist to validate SSH hardening, patching, firewall rules, and removal of unnecessary services.

5. Network Security Assessment with nmap (Workstation -> Server)

To evaluate network exposure, I ran nmap scans from the Workstation VM to the Server VM using the Host-Only IP. This allowed me to identify open ports and confirm which services were reachable over the internal lab network. I focused on safe scanning options suitable for a student lab environment.

Install nmap and confirm version (Workstation VM):

```
sudo apt update
sudo apt install -y nmap
nmap --version | tee ~/week7_audit/nmap_version.txt
```

Connectivity test to server:

```
ping -c 4 <SERVER_IP> | tee ~/week7_audit/ping_server.txt
```

Scan commands used (replace <SERVER_IP> with the server Host-Only IP):

```
nmap -sS -Pn <SERVER_IP> | tee ~/week7_audit/nmap_basic.txt
nmap -sV -Pn <SERVER_IP> | tee ~/week7_audit/nmap_services.txt
nmap -sC -sV -Pn <SERVER_IP> | tee ~/week7_audit/nmap_default_scripts.txt
```

I captured screenshots of the scan results and recorded which ports were open. For each detected open port, I linked it to a running service on the server (checked using ss -tulpn) and justified why it is required. If a port was unexpected, I investigated the owning process and considered disabling it.

Open ports and justification (fill based on your nmap output):

Port	Proto	Detected service	State	Risk / notes	Justification / action
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access / disabled if not needed]
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access / disabled if not needed]
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access / disabled if not needed]
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access /

					disabled if not needed]
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access / disabled if not needed]
[e.g., 22]	[tcp/udp]	[e.g., ssh]	[open/filtered]	[low/medium/high + why]	[required for admin access / disabled if not needed]

6. Access Control Verification (Server VM)

I verified access control to ensure the server follows least privilege. I checked who can use sudo, how authentication is configured, and whether SSH allows risky options such as root login or password authentication. These checks support accountability and reduce the chance of unauthorized access.

Identity and sudo checks:

```
whoami
id
getent group sudo | tee ~/week7_audit/sudo_group_members.txt
sudo -l | tee ~/week7_audit/sudo_privileges.txt
```

Password policy checks:

```
sudo cat /etc/login.defs | egrep "PASS_MAX_DAYS|PASS_MIN_DAYS|PASS_WARN_AGE" | tee
~/week7_audit/password_policy_login_defs.txt
sudo cat /etc/pam.d/common-password | tee ~/week7_audit/pam_common_password.txt
```

Effective SSH policy checks:

```
sudo sshd -T | egrep
"permitrootlogin|passwordauthentication|pubkeyauthentication|x11forwarding|allowusers|allowgroups" | tee
~/week7_audit/ssh_effective_config.txt
```

Access control results summary (fill with what your system showed):

Check	Command / evidence	Observed result	Why it matters / actions
Users with sudo	getent group sudo	[list usernames]	Only trusted admin accounts should be in sudo group.
Sudo permissions	sudo -l	[allowed commands]	Confirms privileges are appropriate and auditable.
Root SSH login	sshd -T | permitrootlogin	[yes/no]	Root SSH login should be disabled to reduce brute-force risk.
SSH password authentication	sshd -T | passwordauthentication	[yes/no]	Key-based auth is safer than passwords in most cases.
Password aging policy	login.defs PASS_*	[values]	Ensures passwords expire and warnings are

			shown.
Password complexity	pam common-password	[policy summary]	Helps resist weak password attacks.

7. Service Audit and Justification (Server VM)

A key part of hardening is reducing the attack surface. I performed a service audit to identify every running service and confirm that each one is necessary for my server. I used systemctl to list running services and ss to map listening ports to processes. Any services not required for the server role were candidates for disabling.

Commands used to list running services and listening ports:

```
systemctl list-units --type=service --state=running | tee ~/week7_audit/running_services.txt
sudo ss -tulpn | tee ~/week7_audit/listening_ports.txt
```

Running service justification table (fill based on your running_services.txt):

Service	Purpose	Related port (if any)	Justification (why needed)	Action taken
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]
[service name]	[what it does]	[e.g., 22/tcp or none]	[why it is required on this server]	[kept / disabled / masked]

If I identified an unnecessary service, I used the following approach to stop and prevent it from starting again:

```
sudo systemctl disable --now <service-name>
# Re-check
systemctl list-units --type=service --state=running
sudo ss -tulpn
```

8. System Configuration Review (Server VM)

After scanning and auditing services, I reviewed key configuration areas that affect security: firewall rules, automatic updates, mandatory access control, intrusion prevention, and logging. This helps confirm the system is hardened and monitored, not just 'clean' on a scan.

Firewall status:

```
sudo ufw status verbose | tee ~/week7_audit/ufw_status.txt
# If using nftables instead:
# sudo nft list ruleset | tee ~/week7_audit/nft_ruleset.txt
```

Automatic updates (unattended-upgrades):

```
dpkg -l | egrep "unattended-upgrades|apt-listchanges" | tee ~/week7_audit/auto_updates_packages.txt
systemctl status unattended-upgrades --no-pager | tee ~/week7_audit/unattended_upgrades_status.txt
```

Mandatory Access Control (AppArmor) status:

```
sudo aa-status | tee ~/week7_audit/apparmor_status.txt
```

Fail2ban status (if installed):

```
sudo systemctl status fail2ban --no-pager | tee ~/week7_audit/fail2ban_status.txt
sudo fail2ban-client status | tee ~/week7_audit/fail2ban_overview.txt
```

Configuration review summary (fill with what your system showed):

Area	Evidence	Status / result	Notes / improvements
Firewall	ufw status verbose (or nft ruleset)	[enabled/disabled + allowed ports]	Only required ports should be allowed; block everything else.
Auto updates	unattended-upgrades status	[active/inactive]	Automatic updates reduce exposure to known vulnerabilities.
AppArmor	aa-status	[enforcing/profiles loaded]	MAC adds an extra security layer beyond standard file permissions.
Fail2ban	fail2ban status	[active/inactive + jails]	Helps mitigate brute-force attempts against services like SSH.
Logging	journalctl and log files	[reviewed]	Logs support detection and incident response.

9. Overall Evaluation and Conclusions

Based on the Lynis hardening score and the manual verification steps, I evaluated the system as follows:

- What is strong in my current configuration:
    - I can clearly see which ports are exposed using nmap and ss, and I can justify why they are open.
    - I verified sudo membership and SSH policies to support least privilege and secure administration.
    - I reviewed firewall configuration and update settings to reduce vulnerability exposure over time.
    - I confirmed mandatory access control status (AppArmor) to improve containment of services.
- What I will improve next (based on Lynis warnings/suggestions):
    - Apply any high-impact Lynis recommendations related to SSH hardening and authentication.

- Remove or disable any non-essential services identified in the service audit.
- Ensure automatic security updates are enabled and monitored.
- Review kernel/network hardening settings suggested by Lynis where appropriate for a student lab.

This week's audit gave me a clear view of the server's security posture. By combining automated scanning (Lynis) with network scanning (nmap) and manual configuration checks, I was able to validate that the system configuration matches a safer baseline and is easier to defend and maintain.

10. Evidence Checklist (Screenshots and Saved Outputs)

For submission, I included the following evidence items. Each item has a corresponding screenshot or a saved output file in ~/week7_audit.

- Server VM: ip addr showing Host-Only IP address
- Workstation VM: ip addr showing Host-Only IP address
- Lynis: installation/version output and final summary screen (hardening index, warnings, suggestions)
- Lynis: saved output files (lynis_audit.txt, lynis.log, lynis-report.dat, lynis_key_findings.txt)
- nmap: ping_server.txt showing connectivity
- nmap: scan outputs showing open ports and service versions (nmap_basic.txt, nmap_services.txt, nmap_default_scripts.txt)
- Server VM: running_services.txt and listening_ports.txt (systemctl and ss -tulpn)
- Server VM: access control outputs (sudo_group_members.txt, sudo_privileges.txt, ssh_effective_config.txt)
- Server VM: firewall status output (ufw_status.txt or nft_ruleset.txt)
- Server VM: AppArmor status output (apparmor_status.txt)
- Server VM: unattended-upgrades status output (unattended_upgrades_status.txt)
- Server VM: fail2ban status output if installed (fail2ban_status.txt, fail2ban_overview.txt)