Week 9 Journal: System Security and Hardening

Course: Operating Systems **Topic**: Security Auditing, Network Assessment, Hardening, and Best Practices **Instructors**: Dr Shabih Fatima & Tanaya Bowade

---

Table of Contents

---

1. Introduction to Security Auditing

What is a Security Audit?

A security audit is a systematic evaluation of an organization's security infrastructure, policies, and practices to identify vulnerabilities before they can be exploited.

Key Functions: - Assesses controls and ensures compliance - Detects breaches and recommends improvements - Evaluates security across

various aspects

Security Audit vs. Other Security Assessments

Feature	Security Audit	Vulnerability Assessment	Penetration Testing
Focus	Comprehensive evaluation	Identifying weaknesses	Exploiting vulnerabilities
Approach	Systematic review	Scanning for flaws	Active exploitation
Techniques	Inspection, testing	Automated scanning	Simulated attacks

Key Insight: Security auditing is a critical component of organizational security that provides a holistic view of the security posture.

---

2. Network Security Assessment Principles

Assessment Methods

1. Penetration Testing

- Simulates real-world attacks to identify exploitable weaknesses
- Active testing methodology
- Helps understand actual risk exposure

2. Vulnerability Scanning

- Systematically identifies known security flaws and misconfigurations
- Automated approach
- Regular scanning is essential

Assessment Tools

Tool	Purpose

Tool	Purpose
Nmap	Network discovery and auditing
Nessus	Vulnerability scanning
Wireshark	Packet analysis
Metasploit	Exploitation framework

Port Scanning with Nmap Example

`bash $ nmap -sS 192.168.1.100` This performs a SYN scan to detect open ports without completing the TCP handshake.

Assessment Workflow

1. Reconnaissance - Gathering information about target network and systems - Uses both passive and active techniques

2. Enumeration - Identifying live hosts, open ports, services, and OS versions - Maps the attack surface

3. Analysis - Evaluating identified services for vulnerabilities - Assessing misconfigurations and potential exploits

Systematic Approach Components

Asset Discovery

- Identify all devices on the network `bash $ nmap -sn 192.168.1.0/24`

Vulnerability Scanning

- Identify security weaknesses in systems
- Tools: Nessus, Qualys, OpenVAS

Port Analysis

- Identify open ports and services `bash $ nmap example.com`

Risk Prioritization

- Prioritize risks by impact and likelihood
- Use risk matrix to plot vulnerabilities

---

3. Security Auditing with Automated Tools

Overview

Automated security auditing identifies weaknesses before exploitation, providing proactive security measures.

Key Auditing Tools

1. Lynis

- Comprehensive scanner for security hardening and vulnerabilities
- Provides hardening index score
- Generates detailed recommendations

2. OpenSCAP

- SCAP (Security Content Automation Protocol) implementation
- Used for security policy auditing
- Standards-based compliance checking

3. chkrootkit

- Rootkit detection tool
- Scans for known rootkit signatures
- Essential for compromise detection

Lynis Audit Example

```bash Lynis 3.0.8

---
```

System details - Distribution: Ubuntu 22.04 LTS - Kernel: Linux 5.15.0

Security Essentials - Firewall: [���] UFW (active) - SSH: [���] OpenSSH (running) - Malware scanner: [X] Not found

Hardening Index : 75/255

Warnings (1) - KRNL-5820: Found unbootable kernel entries

Suggestions (5) - AUTH-9286: Enable two-factor authentication ```

Audit Workflow

1. **Scan System** - Run audit tools
2. **Generate Report** - Review findings
3. **Implement Fixes** - Address issues
4. **Re-scan** - Verify fixes

This iterative approach ensures continuous security improvement.

---

4. Security Hardening Strategies

Definition

Security hardening is the process of reducing a system's attack surface by eliminating potential vulnerabilities and strengthening defenses.

Core Hardening Strategies

1. Disable Unused Services

- Identify and disable unnecessary services
- Disable legacy protocols (Telnet, HTTP)
- Use systemctl to mask services not needed

Example: `bash systemctl mask telnet.service`

2. Keep Systems Updated

- Regularly apply security patches
- Enable automatic updates for critical packages
- Test updates in staging before production

Example: `bash sudo apt update && sudo apt upgrade`

3. Implement Mandatory Access Control

- Configure SELinux with targeted policy
- Implement AppArmor profiles for applications
- Use least privilege for process permissions

4. Enforce Strong Authentication

- Use SSH keys instead of passwords
- Implement two-factor authentication
- Configure password policies in PAM

Principle of Least Privilege

Grant only minimum necessary permissions to users and processes.

Example: A user should only have access to web server content directory, not system configuration files.

Comprehensive Patch Management

Systematically identify, test, and apply security updates to all systems.

Linux Example: `bash sudo apt update && sudo apt upgrade`

Regular monitoring and testing is essential.

---

5. Linux Security Hardening Best Practices

Overview

Linux powers a significant portion of the internet's infrastructure. Implementing proper security measures is critical.

Key Best Practices

1. File Permissions

Control access to files and directories - Three categories: User, Group, Others - Read (r), Write (w), Execute (x) permissions

Example: `bash chmod 600 sensitive_file.txt`

2. Advanced Attributes (chattr)

Provides extra layer of security - Immutable attribute (+i) prevents modification - Effective against accidental changes and malicious modifications

Example: `bash sudo chattr +i /etc/passwd`

3. SSH Hardening

Secure remote access - Disable root login - Use key-based authentication - Change default SSH port

Example Configuration: `bash PasswordAuthentication no`

Important Principle: "Security is a process, not a product"

---

6. File Permissions and Advanced Attributes

Linux File Permissions

Linux permissions control access to files and directories.

Permission Categories

- **User (u)** - Owner permissions
- **Group (g)** - Permissions for group
- **Others (o)** - Permissions for everyone else

Permission Types

- **r** - Read (4)
- **w** - Write (2)
- **x** - Execute (1)

Numeric Notation Example

`6 = User: Read (4) + Write (2) 0 = Group: No permissions 0 = Others: No permissions`

Setting Permissions: `bash chmod 600 sensitive_file.txt`

Advanced Attributes (chattr)

The chattr command provides advanced file attributes for enhanced security.

Immutable Attribute (+i)

Prevents file modification, deletion, or renaming.

Important: Even root cannot modify immutable files without removing the attribute first.

Examples: ```bash

Set immutable attribute

$ sudo chattr +i /etc/passwd

Remove immutable attribute

$ sudo chattr -i /etc/passwd ```

Key Takeaway: Proper permission management is critical for system security.

---

7. User and Access Management

Secure User Policies

1. Password Aging

- Enforces password expiration policies
- Uses the chage command
- Ensures regular password updates

2. PAM Configuration

- Pluggable Authentication Modules
- Flexible authentication framework for enforcing policies
- Centralized authentication management

3. Restrictive Sudo Policy

- Configured in /etc/sudoers via visudo
- Grants specific privileges to specific users
- Prevents unauthorized administrative access

4. Least Privilege Principle

- Grant only minimum necessary permissions
- Reduces attack surface
- Limits damage from compromised accounts

Access Control Examples

Restricting Root Access via SSH

```bash

# Edit /etc/ssh/sshd_config

PermitRootLogin no

# Restart SSH service

sudo systemctl restart sshd ```

**Purpose:** Prevents direct root login over SSH

**Reviewing Sudo Logs**

```bash

# Check sudo logs

grep sudo /var/log/auth.log ```

**Purpose:** Provides audit trail of administrative actions

### Roles vs. Privileges Example

| Role | Description | Sudo Privileges |
| --- | --- | --- |
| Web Admin | Manages web server | Restart web service |

This role-based approach ensures users have only the permissions they need.

---

## 8. SSH Security Configuration

### Overview

Secure Shell (SSH) is a primary method for remote access to Linux servers, making it a frequent target for attackers.

## Three Pillars of SSH Security

### 1. Disable Root Login

**Purpose:** Prevent direct root login to avoid immediate superuser privileges upon successful compromise.

**Configuration:** ```bash

# /etc/ssh/sshd_config

PermitRootLogin no ```

**Best Practice:** Users should log in with a regular account and use sudo for administrative tasks.

### 2. Use Key-Based Authentication

**Purpose:** Cryptographic key pairs are more secure than password authentication, resistant to brute-force attacks.

**Configuration:** ```bash

# /etc/ssh/sshd_config

PasswordAuthentication no PubkeyAuthentication yes ```

**Important:** Disable password authentication once key-based authentication is set up.

### 3. Change Default SSH Port

**Purpose:** Changing the default port (22) reduces the volume of automated attacks, though it does not eliminate determined attackers.

**Configuration:** ```bash

# /etc/ssh/sshd_config

Port 2222 ```

**Remember:** Update any firewall rules to allow traffic on the new port.

## Applying SSH Configuration Changes

After making changes to sshd_config, restart the SSH service: `bash sudo systemctl restart sshd`

**Key Principle:** Proper SSH configuration is critical for server security.

---

# 9. Firewall Configuration and Network Security

## Firewall Fundamentals

A network security system that monitors and controls incoming and outgoing traffic based on predefined security rules.

**Key Functions:** - Barrier between trusted/untrusted networks - Filters traffic based on policies - Controls access to resources - Prevents unauthorized access

## Firewall Tools

### 1. iptables

- Traditional, flexible utility for Netfilter
- Complex syntax

- Low-level control

**2. firewalld**

- Dynamic management with D-Bus interface
- Uses zones and services
- Modern approach

**3. ufw (Uncomplicated Firewall)**

- User-friendly frontend for iptables
- Intuitive commands
- Recommended for beginners

## Input/Output Chains

**INPUT Chain**

- Processes packets destined for local system
- Controls what comes into the server

**OUTPUT Chain**

- Processes packets originating from local system
- Controls what leaves the server

## Intrusion Detection

**Fail2Ban**

- Scans logs for malicious patterns
- Automatically updates firewall rules
- Blocks IP addresses after failed attempts

**Snort**

- Network intrusion detection system
- Performs traffic analysis

- Real-time alerting

---

# 10. Monitoring and Incident Response

## Continuous Monitoring

### Log Analysis Tools

**1. journalctl** - Systemd journal logs - Modern logging system

**2. syslog** - General system messages - Traditional logging

**3. auditd** - System call auditing - Detailed security events

### Real-time Alerting Systems

**1. OSSEC** - Host-based IDS - Multi-platform support

**2. Wazuh** - Comprehensive security platform - Built on OSSEC foundation

## Incident Response Process

**1. Detect** - Identify incident **2. Contain** - Limit impact **3. Eradicate** - Remove cause **4. Recover** - Restore service

## Log Snippet Example

``` Oct 26 10:30:05 server sshd[12345]: Failed password for invalid user guest from 192.168.1.100 port 54321 ssh2

Oct 26 10:30:08 server sshd[12346]: Failed password for root from 192.168.1.101 port 54322 ssh2

Oct 26 10:30:11 server sshd[12347]: Failed password for invalid user admin from 192.168.1.100 port 54323 ssh2 ```
```

Analysis: This snippet shows failed SSH login attempts, which could trigger an alert indicating a potential brute-force attack.

---

11. Integrating Security into System Design

Secure Architecture Principles

OS-level security is a foundational component of system architecture

Key Components

1. Secure Boot - Ensures only trusted software can load during boot process - Cryptographic verification - Hardware-level security

2. Kernel Module Integrity - Verifies loaded kernel modules haven't been tampered with - Prevents rootkit installation - Maintains system integrity

Automation & Consistency

Configuration Management

- Ansible, Puppet for uniform security policy enforcement
- Infrastructure as Code
- Consistent deployment

Automated Backups

- Regular backups protect against data loss
- Disaster recovery capability
- Business continuity

Automated Patching

- Ensures systems are promptly updated with security fixes
- Reduces vulnerability window
- Systematic approach

Secure Deployment Example (Layered Security)

Application Layer - Web Application Firewall

Web Server Layer - Nginx/Apache with TLS/SSL

Firewall Layer - iptables/firewalld

Hardened OS Layer - SELinux/AppArmor, Secure Boot

Hardware Layer - TPM, Secure Boot BIOS

Key Takeaway: Security is most effective when integrated into the system architecture from the ground up, not added as an afterthought.

---

12. Practical Troubleshooting Scenario

Scenario: New Sysadmin First Day Assessment

You're the new sysadmin for a small web development company. On your first day, you run Lynis and discover:

Security Issues Found: - Hardening score: 68/255 - Root login via SSH is enabled - Password authentication is enabled - The /etc/passwd file has 644 permissions (world-readable) - No sudo logging is configured - Three developers share the same 'devops' account with root privileges

Auth Log Evidence: `Failed password for root from 185.220.101.50 port 45123 ssh2 Failed password for root from 185.220.101.50 port 45124 ssh2 Failed password for root from 185.220.101.50 port 45125 ssh2 [repeats 47 more times in the last hour]`

Critical Security Issues Analysis

Task: Using the concepts covered (file permissions, chattr, SSH hardening, sudo policies), identify THREE most critical security issues and propose a specific fix for each.

Issue 1: Root Login via SSH Enabled + Brute Force Attack

Risk: Active brute-force attack attempting to compromise root account

Fix: ```bash

1. Disable root login

echo "PermitRootLogin no" >> /etc/ssh/sshd_config

2. Disable password authentication

echo "PasswordAuthentication no" >> /etc/ssh/sshd_config

3. Enable key-based authentication only

echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config

4. Restart SSH

sudo systemctl restart sshd

5. Install and configure fail2ban

sudo apt install fail2ban ```

Issue 2: World-Readable /etc/passwd File

Risk: Sensitive user information exposed

Fix: ```bash

Correct permissions

sudo chmod 644 /etc/passwd # Actually this is standard

The real concern is /etc/shadow

sudo chmod 600 /etc/shadow

Make immutable

sudo chattr +i /etc/passwd sudo chattr +i /etc/shadow ```

Issue 3: Shared 'devops' Account with Root Privileges

Risk: No accountability, shared credentials, excessive privileges

Fix: ```bash

1. Create individual accounts for each developer

sudo useradd -m -s /bin/bash dev1 sudo useradd -m -s /bin/bash dev2 sudo useradd -m -s /bin/bash dev3

2. Configure restrictive sudo with logging

sudo visudo

Add specific permissions for each user, not full root

3. Enable sudo logging

echo "Defaults logfile=/var/log/sudo.log" | sudo tee -a /etc/sudoers

4. Disable the shared devops account

sudo usermod -L devops ```

---

13. Assessment Week 5: Advanced Security and Monitoring

Phase 5: Advanced Security and Monitoring Infrastructure

Objective: Implement advanced security controls and develop monitoring capabilities.

Deliverables (Journal and Video)

1. Implement Access Control using SELinux or AppArmor

Requirements: - Configure mandatory access control system - Document how to track and report on access control settings - Show enforcement policies

SELinux Example: ```bash

Check SELinux status

sestatus

Set to enforcing mode

sudo setenforce 1

Make permanent

sudo vi /etc/selinux/config

SELINUX=enforcing

View current context

ls -Z /var/www/html ```

AppArmor Example: ```bash

Check status

sudo apparmor_status

Create profile

sudo aa-genprof /usr/bin/application

Enforce profile

sudo aa-enforce /etc/apparmor.d/usr.bin.application ```

2. Configure Automatic Security Updates

Requirements: - Enable unattended upgrades - Provide evidence of implementation - Configure update policies

Implementation: ```bash

Install unattended-upgrades

sudo apt install unattended-upgrades

Enable automatic updates

sudo dpkg-reconfigure -plow unattended-upgrades

Configure

sudo vi /etc/apt/apt.conf.d/50unattended-upgrades ```

Evidence: ```bash

Check logs

sudo cat /var/log/unattended-upgrades/unattended-upgrades.log ```

3. Configure fail2ban for Enhanced Intrusion Detection

Requirements: - Install and configure fail2ban - Set up jail for SSH - Configure ban policies

Implementation: ```bash

Install fail2ban

sudo apt install fail2ban

Create local configuration

sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

Configure SSH jail

sudo vi /etc/fail2ban/jail.local

[sshd]

enabled = true

port = 22

maxretry = 3

bantime = 3600

Start service

sudo systemctl start fail2ban sudo systemctl enable fail2ban

Check status

sudo fail2ban-client status sshd ```

4. Create Security Baseline Verification Script

File: `security-baseline.sh`

Requirements: - Runs on the server (executed via SSH) - Verifies all security configurations from Phases 4 and 5 - Include line-by-line comments

Script Structure: ```bash

!/bin/bash

Security Baseline Verification Script

Verifies all security configurations

Check SSH configuration

Check firewall status

Verify user permissions

Check for unauthorized SUID files

Verify SELinux/AppArmor status

Check automatic updates configuration

Verify fail2ban status

Generate compliance report

```

**5. Create Remote Monitoring Script**

**File:** `monitor-server.sh`

**Requirements:** - Runs on workstation - Connects via SSH - Collects performance metrics from server - Include line-by-line comments

**Script Structure:** ```bash

# !/bin/bash

# Remote Server Monitoring Script
```

Connects via SSH and collects metrics

Connect to server via SSH

Collect CPU usage

Collect memory usage

Collect disk usage

Check running processes

Monitor network connections

Generate monitoring report

```

### Important Notes

**All scripts must:** - Include line-by-line comments explaining functionality - Be demonstrated in video with live command execution - Include error handling - Generate clear, readable output

**Video must show:** - Live command execution - Clear explanation of each step - Verification of configurations - Demonstration of monitoring capabilities

---
```

Review & Test Preparation

Key Questions to Test Your Knowledge

1. **What is the primary function of Lynis?**

2. Comprehensive security scanner for hardening and vulnerabilities

3. **Explain the difference between SUID and SGID**

4. SUID: Runs with file owner's permissions

5. SGID: Runs with file group's permissions

6. **Which command can display advanced file attributes?**

7. `lsattr` displays attributes

8. `chattr` modifies attributes

9. **How does ufw simplify firewall configuration?**

10. User-friendly frontend for iptables
11. Intuitive commands

12. Simplified syntax

13. **Name two principles of security hardening**

14. Principle of Least Privilege
15. Defense in Depth
16. Disable Unused Services

17. Keep Systems Updated

18. **What's the purpose of auditd in Linux?**

19. System call auditing
20. Tracks security-relevant events

21. Provides detailed audit trail

22. **How does SELinux enforce access control?**

23. Mandatory Access Control (MAC)
24. Policy-based enforcement

25. Security contexts and labels

26. **Describe one method to detect rootkits**

27. Use chkrootkit or rkhunter
28. Compare system files against known-good versions
29. Monitor for suspicious kernel modules

Key Takeaways

1. **Security is a Process, Not a Product**
2. Continuous monitoring and improvement required

3. Regular audits essential

4. **Defense in Depth**

5. Multiple layers of security controls

6. No single point of failure

7. **Principle of Least Privilege**

8. Grant minimum necessary permissions

9. Reduces attack surface

10. **Automation is Critical**

11. Consistent configuration management
12. Automated patching and updates

13. Reduces human error

14. **Monitoring and Response**

15. Continuous monitoring essential
16. Incident response plan required

17. Log analysis critical

18. **Security by Design**

19. Integrate security from the start
20. Not an afterthought
21. Architecture-level considerations

References

- Week 09 System Security and Hardening.pdf
- Security Auditing with Automated Tools v2.pdf
- Assessment Week 5.pdf
- Linux Security Hardening and Best Practices

End of Week 9 Journal

Table of Contents