Week 7 Journal: Initial Security Configuration

Course: Operating System Security Fundamentals **Instructors:** Tanaya Bowade & Dr Shabih Fatima **Date:** Week 7

Table of Contents

OS Security Fundamentals

The AAA Framework

Operating System security fundamentally relies on the "AAA" framework, which ensures that only legitimate users can access resources and that their actions are restricted according to defined policies.

1. Authentication

- **Definition:** Verifies the identity of a user or process

- **Question Answered:** "Who are you?"
- **Methods:**
- Usernames and passwords
- Biometrics
- Digital certificates
- **Purpose:** Ensures that the entity attempting to access the system is who they claim to be

2. Authorization

- **Definition:** Determines what an authenticated user or process is permitted to do
- **Question Answered:** "What can you do?"
- **Implementation:** Involves defining permissions for specific resources or actions
- **Purpose:** Restricts users to only the resources and operations they need

3. Access Control

- **Definition:** Enforces the authorization policies
- **Mechanism:** The system actively grants or denies requests based on established authorizations
- **Purpose:** Prevents unauthorized access or actions by ensuring rules are followed
- **Function:** This is where the system actively implements the security policies

Key Concept

The AAA framework works together as a comprehensive security approach: 1. First, **authenticate** the user (verify identity) 2. Then, **authorize** what they can do (check permissions) 3. Finally, **control access** to enforce those permissions (implement the rules)

Importance of Initial Security Setup

A robust initial security configuration is paramount to prevent common vulnerabilities that attackers frequently exploit. Proper initial security configuration is the first line of defense against these common vulnerabilities.

Common Vulnerabilities

1. Default Passwords

- **Description:** Pre-set, easily guessable passwords (e.g., "admin," "password")
- **Risk:** Provide direct entry points for unauthorized access
- **Example:** Router with default password "admin123"
- **Mitigation:** Change all default passwords immediately upon system installation

2. Unnecessary Services

- **Description:** Default installations include many services that are not essential
- **Risk:** Each service represents a potential attack surface
- **Impact:** More services = more potential vulnerabilities
- **Mitigation:** Disable all unnecessary services and only enable what's required

3. Improper Permissions

- **Description:** Incorrect file and directory permissions
- **Risk:** Allow unauthorized users to access sensitive system files
- **Impact:** Can lead to data breaches or system compromise
- **Mitigation:** Implement principle of least privilege for all file permissions

Attack Vectors

Network Exploits

- **Method:** Default passwords or open network services exploited remotely
- **Result:** Attackers gain initial system access
- **Prevention:**
- Change default credentials
- Close unnecessary ports
- Implement firewall rules

Local Privilege Escalation

- **Method:** Improper file permissions or misconfigured services
- **Result:** Low-privileged user gains higher-level access
- **Prevention:**
- Proper permission management
- Regular security audits
- Principle of least privilege

OS-Level Vulnerabilities & Impact

Understanding common vulnerabilities and their impact on system security is crucial for maintaining a secure operating system.

Common OS Vulnerabilities

1. Buffer Overflows

- **Description:** Writing data beyond allocated buffer size, potentially overwriting adjacent memory
- **Impact:** System Crash
- **Example:** Heartbleed bug in OpenSSL
- **Risk:** Can lead to arbitrary code execution
- **Consequences:**
- System crashes
- Unauthorized code execution
- Memory corruption

2. Race Conditions

- **Description:** Dependencies on sequence or timing of events
- **Impact:** Privilege Escalations
- **Risk:** Can lead to privilege escalation if events don't happen in expected order
- **Mechanism:** Exploits timing vulnerabilities between operations
- **Example:** Time-of-check to time-of-use (TOCTOU) vulnerabilities

3. Improper Input Validation

- **Description:** Failure to sanitize user input
- **Impact:** Data Leak
- **Risk:** Can lead to injection attacks (SQL, command injection) or buffer overflows
- **Types of Attacks:**
- SQL injection
- Command injection
- Cross-site scripting (XSS)
- Buffer overflow attacks

4. Use-After-Free Vulnerabilities

- **Description:** Using memory after it has been freed
- **Impact:** System Crash
- **Risk:** Can lead to data corruption, crashes, or arbitrary code execution
- **Exploitation:** Attackers can manipulate freed memory for malicious purposes

System Impact

System Integrity

Definition: Assurance that information is not altered or destroyed in an unauthorized manner

When Compromised: - Data can be modified, corrupted, or deleted - Unreliable system operations - Persistent backdoors or service disruptions - Loss of trust in system data - Potential for long-term damage

Confidentiality

Definition: Ensuring sensitive information is accessed only by authorized individuals

When Breached: - Private data exposed to unauthorized parties - Identity theft or corporate espionage - Legal and reputational consequences - Regulatory compliance violations - Financial losses

---

Security Models: DAC, MAC, RBAC

Access control models define how subjects (users, processes) can access objects (files, resources). Three primary models are widely used in modern operating systems.

Comparison Table

Feature	Discretionary Access Control (DAC)	Mandatory Access Control (MAC)	Role-Based Access Control (RBAC)
Control Method	Owner or authorized user grants/revokes permissions	Central authority defines and enforces security labels and rules	Permissions assigned to roles; users assigned to roles
Flexibility	High; owners have significant control over their resources	Low; strict, system-wide policies override user discretion	Moderate to High; flexible role definitions, but policies are centralized
Typical Use Case	Personal computing, less sensitive corporate environments	High-security systems (e.g., military, government, classified data)	Enterprise systems, large organizations, compliance-driven environments

1. Discretionary Access Control (DAC)

Characteristics: - Owner-centric permissions - Resource owners control access to their files - High flexibility - User discretion in granting permissions

Advantages: - Easy to implement and manage - Users have control over their own resources - Flexible and intuitive

Disadvantages: - Vulnerable to user errors - Permissions can be changed by malicious owners - Less secure for sensitive environments

Examples: - Unix/Linux file permissions - Windows NTFS permissions

2. Mandatory Access Control (MAC)

Characteristics: - System-enforced security levels - Central authority controls all access decisions - Low flexibility - Strict, system-wide policies

Advantages: - High security level - Robust against user errors and insider threats - Consistent policy enforcement

Disadvantages: - Complex to implement and manage - Less flexible for users - Requires careful planning and configuration

Examples: - SELinux (Security-Enhanced Linux) - AppArmor - Military/government systems

3. Role-Based Access Control (RBAC)

Characteristics: - Role-privilege mapping - Users assigned to roles - Moderate to high flexibility - Centralized policy management

Advantages: - Scalable for large organizations - Easier to manage than individual permissions - Aligns with organizational structure

Disadvantages: - Can become complex with many roles - Requires careful role design - Role explosion in large systems

Examples: - Enterprise systems - Database management systems - Cloud platforms (AWS IAM, Azure RBAC)

Mandatory Access Control (MAC) Concepts

Mandatory Access Control (MAC) is a security model where access decisions are made by a central authority based on security labels assigned to subjects and objects, providing a high level of security by strictly enforcing information flow policies.

Key Components

1. Reference Monitor

- **Location:** Core component of the OS kernel
- **Function:** Mediates all access attempts between subjects and objects
- **Responsibility:** Enforces security policies by comparing security labels and making access decisions
- **Characteristics:**
- Tamper-proof
- Always invoked
- Small enough to be analyzed and tested
- Cannot be bypassed

Access Decision Process: 1. Subject (User A) with security label "Top Secret" attempts to access Object (File X) 2. Request goes through Reference Monitor (OS Kernel) 3. Reference Monitor compares security labels 4. Access Decision (Allow/Deny) is made based on security policy

2. Security Labels

- **Assignment:** System-defined labels assigned by administrators
- **Purpose:** Classify subjects and objects
- **Function:** Represent sensitivity levels and clearance requirements
- **Characteristics:**
- Cannot be changed by users
- Hierarchical or categorical
- Determine what accesses are allowed
- Enforced at the kernel level

Common Security Levels: - Top Secret - Secret - Confidential - Unclassified

Implementation Examples

SELinux (Security-Enhanced Linux)

- **Type:** Policy-driven approach
- **Mechanism:** Type enforcement, role-based access control, multi-level security
- **Use Case:** Enterprise servers, government systems
- **Complexity:** High, but very flexible and powerful
- **Policy Types:** Targeted, MLS (Multi-Level Security), strict

AppArmor

- **Type:** Path-based security
- **Mechanism:** Application-level mandatory access control
- **Use Case:** Desktop systems, servers
- **Complexity:** Simpler than SELinux
- **Approach:** Profile-based confinement

DAC vs. MAC Comparison

Key differences between Discretionary and Mandatory Access Control models:

Feature	Discretionary Access Control (DAC)	Mandatory Access Control (MAC)
Control Method	Owner-centric; resource owner defines access permissions	System-centric; OS enforces access based on security labels/policies
Flexibility	High; owners can easily modify permissions	Low; policies are centrally managed and difficult for users to override

Feature	Discretionary Access Control (DAC)	Mandatory Access Control (MAC)
Management	Decentralized; managed by individual resource owners	Centralized; managed by system administrators or security policies
Security Level	Lower; susceptible to user errors and malicious owners	Higher; robust against user errors and insider threats
Policy Source	User-defined permissions (e.g., file permissions)	System-wide security policy (e.g., security labels, clearance levels)
Example Systems	Unix/Linux file permissions, Windows NTFS permissions	SELinux, AppArmor, military/government systems

When to Use DAC

- Personal computing environments
- Small organizations with trusted users
- Systems with less sensitive data
- Environments requiring user flexibility

When to Use MAC

- Military and government systems
- High-security environments
- Systems handling classified information
- Environments with strict compliance requirements
- Protection against insider threats

Introduction to Intrusion Detection Systems

An Intrusion Detection System (IDS) is a security technology that monitors network or system activities for malicious activities or policy violations and produces reports to a management station. Its primary role is to detect unauthorized access, misuse, or denial-of-service attacks.

Types of IDS

1. Host-based IDS (HIDS)

Monitoring Focus: Monitors activities on a specific host or endpoint

Data Sources: - System logs - File system changes - Process activities

Detection Capabilities: - Internal attacks - Unauthorized file access - Privilege escalation - System configuration changes - Local security policy violations

Example: A security agent installed on a server that monitors local activities and logs

Advantages: - Detailed view of host activities - Can detect attacks that network IDS might miss - Monitors encrypted traffic (after decryption)

Disadvantages: - Resource intensive on the host - Only monitors the local system - Can be disabled if host is compromised

2. Network-based IDS (NIDS)

Monitoring Focus: Monitors network traffic on a network segment

Data Sources: - Network packets - Headers and payloads in real-time - Protocol analysis

Detection Capabilities: - Port scans - DoS attacks - Unauthorized protocol usage - Network-based exploits - Malware propagation

Example: A sensor deployed on a network segment that analyzes traffic for attack signatures

Advantages: - Monitors entire network segment - Difficult for attackers to detect - No impact on host performance

Disadvantages: - Cannot see encrypted traffic - May miss host-level attacks - Can be overwhelmed by high traffic

---

IDS Components & Workflow

Detection Techniques

1. Signature-based Detection

Concept: Identifies known attack patterns or signatures

How it Works: - Compares network traffic or system logs against a database of known attack signatures - Pattern matching against predefined rules - Similar to antivirus signature detection

Advantages: - High accuracy for known threats - Low false positives - Fast detection - Easy to understand and implement

Disadvantages: - Ineffective against unknown or polymorphic attacks - Requires constant signature updates - Cannot detect zero-day exploits - Must maintain large signature database

Best For: - Detecting known attacks - Compliance requirements - Production environments

2. Anomaly-based Detection

Concept: Detects deviations from established normal behavior

How it Works: - Builds a baseline of normal system activity - Uses statistical analysis or machine learning - Flags deviations as suspicious - Learns normal behavior patterns

Advantages: - Can detect novel or zero-day attacks - Identifies previously unknown threats - Adapts to environment

Disadvantages: - Can generate high false positives if baseline is not accurate - Requires training period - More complex to implement - Sensitive to changes in normal behavior

Best For: - Detecting zero-day attacks - Research environments - Advanced threat detection

IDS Workflow

1. **Data Collection**
2. Gather data from network traffic or system logs
3. Capture packets or monitor system events

4. Collect relevant security information

5. **Analysis Engine**

6. Process collected data
7. Apply detection rules or algorithms
8. Compare against signatures or baselines

9. Identify potential threats

10. **Detection**

11. Determine if activity is malicious
12. Evaluate threat level

13. Classify type of attack

14. **Threat Assessment**

15. Is this a real threat or false positive?

16. Decision point in the workflow

17. **Response**

18. **If Threat:** Generate Alert
    - Notify security team
    - Log incident details
    - Trigger automated responses
19. **If No Threat:** Continue Monitor
    - Continue normal monitoring
    - Update baselines if needed

Bash Scripting for System Administration

Automating security tasks for enhanced system protection. Bash scripting automates routine security tasks, enhancing efficiency and consistency.

Automation Tasks

1. User Auditing and Management

- **Purpose:** Automating user account management and auditing privileges
- **Tasks:**
- Create/delete user accounts
- Monitor user login activity
- Audit user privileges
- Generate user activity reports
- **Benefits:** Ensures consistent user management and rapid response to security issues

2. Log File Analysis

- **Purpose:** Parsing system logs to identify security events and failed logins
- **Tasks:**
- Monitor authentication logs
- Detect failed login attempts
- Identify suspicious patterns
- Generate security reports
- **Benefits:** Early detection of security incidents and unauthorized access attempts

3. File Permission Control

- **Purpose:** Enforcing correct file permissions to prevent unauthorized access
- **Tasks:**
- Audit file permissions
- Correct improper permissions
- Monitor permission changes
- Enforce security policies

- **Benefits:** Maintains security posture and prevents unauthorized access

Useful Commands

chmod - Change File Permissions

Purpose: Modify file and directory permissions

Example: `bash chmod 600 secret.txt` **Explanation:** Restricts access to the owner only (read and write for owner, no access for group and others)

Common Permission Modes: - `600`: Owner read/write only - `644`: Owner read/write, others read only - `755`: Owner full access, others read/execute - `700`: Owner full access only

chown - Change File Ownership

Purpose: Change file owner and group

Example: `bash chown user:group file.txt` **Explanation:** Assigns ownership of file.txt to user and group

Use Cases: - Transfer file ownership - Set proper ownership after file creation - Correct ownership issues

grep - Search Text Patterns

Purpose: Search for patterns in files or output

Example: `bash grep "failed password" /var/log/auth.log` **Explanation:** Finds failed login attempts in authentication logs

Security Applications: - Monitor authentication failures - Search for error messages - Identify suspicious patterns - Audit system logs

---

Operating System Modes Review

User vs. Kernel Mode

Feature	User Mode	Kernel Mode
Privileges	Limited access to hardware and critical memory	Full access to all hardware and system memory
Execution	User applications, less privileged instructions	Operating system core, device drivers, privileged instructions
Protection	Isolated from other user processes	Can directly interact with hardware and manage system resources
Failure Impact	Typically crashes only the application	Can crash the entire operating system
Transition	System calls (e.g., I/O requests)	Interrupts, exceptions, system calls

Key Concepts

User Mode

- **Purpose:** Run user applications safely
- **Restrictions:** Cannot directly access hardware or critical system resources
- **Protection:** Applications are isolated from each other and the kernel
- **Security:** Failures are contained to the application
- **Example Operations:** Running a text editor, web browser, or user program

Kernel Mode

- **Purpose:** Execute operating system core functions
- **Privileges:** Full access to all system resources
- **Responsibility:** Manage hardware, memory, and processes
- **Risk:** Errors can crash the entire system
- **Example Operations:** Device driver execution, interrupt handling, system calls

Mode Transition

When User Mode Switches to Kernel Mode: - System calls (requesting OS services) - Hardware interrupts - Exceptions or errors - I/O operations

Process: 1. User application needs OS service 2. Makes system call 3. CPU switches to kernel mode 4. Kernel executes privileged operation 5. Returns result to user mode 6. Application continues execution

---

Lab Activities

Learning Objectives

- Understand process lifecycle and states
- Use command-line tools for process monitoring
- Manage system processes effectively
- Implement coursework security controls

Pre-Lab Preparation

Required video lectures to watch before lab: - Process Management Fundamentals - Understanding System Monitoring Tools - Process States and Lifecycle - Operating System Structure

---

Part 1: Process Fundamentals and Management

Task 1.1: Exploring Process States

Objective: Learn to view and understand process information

Commands and Their Purpose:

1. **View all running processes:** `bash ps aux`

Key Columns: - `USER`: Which user owns the process - `PID`: Process identification number - `%CPU`: CPU usage percentage - `%MEM`: Memory usage percentage - `STAT`: Process state - `COMMAND`: The command that started the process

1. **Compare different process listing formats:** `bash ps -ef` Shows processes in full format with different column arrangement

2. **View real-time process monitoring:** `bash top` Provides dynamic, real-time view of system processes. Press 'q' to quit.

3. **Enhanced monitoring with htop:** `bash sudo apt install htop htop` More user-friendly interface with better visualization

Process States: - `R`: Running or runnable (on run queue) - `S`: Interruptible sleep (waiting for an event) - `D`: Uninterruptible sleep (usually I/O) - `Z`: Zombie (terminated but not reaped by parent) - `T`: Stopped (by job control signal or being traced)

Task 1.2: Process Relationships and Control

1. View process hierarchy: `bash pstree pstree -p # Shows PIDs`

2. Start a background process: `bash sleep 300 & jobs`

3. Practice process control: ```bash sleep 500

Press Ctrl+Z to suspend

jobs bg # Resume in background fg # Bring to foreground ```

4. Practice process termination: ```bash sleep 600 & kill [PID] # Graceful termination

sleep 600 & sleep 600 & killall sleep # Kill all processes with name "sleep" ```

Difference between kill and kill -9: - `kill`: Sends SIGTERM (15) - allows graceful shutdown - `kill -9`: Sends SIGKILL (9) - forces immediate termination, no cleanup

5. Experiment with process priority: ```bash nice -n 10 sleep 400 & top

Observe the NI (nice) value

```

**Process Lifecycle:** 1. **New**: Process is being created 2. **Ready**: Process is ready to run 3. **Running**: Process is executing 4. **Waiting**: Process is waiting for I/O or event 5. **Terminated**: Process has finished execution

**When to use foreground vs background:** - **Foreground**: Interactive tasks requiring user input - **Background**: Long-running tasks that don't need interaction

---

## Part 2: SSH Key-Based Authentication Setup

### Task 2.1: Generating SSH Keys

**Generate SSH key pair:** `bash ssh-keygen -t ed25519 -C "your_email@example.com"`

**Steps:** 1. Accept default location (`~/.ssh/id_ed25519`) 2. Set a secure passphrase 3. Keys are generated: private key (`id_ed25519`) and public key (`id_ed25519.pub`)

**View your public key:** `bash cat ~/.ssh/id_ed25519.pub`

**Why ed25519 is recommended:** - More secure than RSA with smaller key size - Faster key generation and verification - Better resistance to timing attacks - Modern cryptographic algorithm - 256-bit security (equivalent to 3072-bit RSA)

### Task 2.2: Copying Key to Server
```

Copy public key to server: ```bash ssh-copy-id username@server_ip

Enter password when prompted

```

**Test passwordless login:** `bash ssh username@server_ip`

**How it works:** 1. Public key is appended to `~/.ssh/authorized_keys` on server 2. When connecting, server challenges with public key 3. Client proves identity with private key 4. No password needed

### Task 2.3: Hardening SSH Configuration

**1. Backup original configuration:** `bash sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.backup`

**2. Edit SSH configuration:** `bash sudo nano /etc/ssh/sshd_config`

**3. Critical security changes:** `PasswordAuthentication no PubkeyAuthentication yes PermitRootLogin no`

**Security Improvements:** - `PasswordAuthentication no`: Prevents brute-force password attacks - `PubkeyAuthentication yes`: Enables key-based authentication - `PermitRootLogin no`: Prevents direct root access, requiring privilege escalation

**4. Restart SSH service:** `bash sudo systemctl restart sshd`

**5. Verify changes:** Test from different terminal to ensure password authentication is disabled

---

## Part 3: Firewall Configuration and User Management

### Task 3.1: Implementing Firewall Rules

**1. Check UFW status:** `bash sudo ufw status`
```

2. Install UFW if needed: `bash sudo apt update sudo apt install ufw`

3. Set default policies: `bash sudo ufw default deny incoming sudo ufw default allow outgoing`

Why these defaults? - **Deny incoming**: Blocks all unsolicited incoming connections (whitelist approach) - **Allow outgoing**: Permits internal services to communicate externally

4. Allow SSH from specific IP: `bash sudo ufw allow from workstation_ip to any port 22`

Security benefit: SSH access restricted to known workstation only

5. Enable firewall: `bash sudo ufw enable`

6. Verify rules: `bash sudo ufw status numbered sudo ufw status verbose`

Defense-in-Depth Strategy: Multiple layers of security controls: 1. Firewall (network level) 2. SSH key authentication (access level) 3. User privileges (authorization level) 4. File permissions (data level)

Task 3.2: User and Privilege Management

1. Create non-root administrative user: ```bash sudo adduser adminuser

Set strong password when prompted

```

**2. Add user to sudo group:** `bash sudo usermod -aG sudo adminuser`

**3. Verify group membership:** `bash groups adminuser id adminuser`
```

4. Test sudo access: `bash su - adminuser sudo apt update whoami # Should show adminuser`

5. List users with sudo privileges: `bash getent group sudo`

Principle of Least Privilege: - Users should have minimum permissions necessary - Administrative tasks performed with sudo, not as root - Reduces risk of accidental system damage - Provides audit trail of privileged actions - Limits damage from compromised accounts

Why use non-root administrative user: - Prevents accidental system damage - Provides accountability (commands logged with username) - Requires explicit privilege escalation (sudo) - Better security posture - Follows security best practices

Task 3.3: Remote Administration Evidence

Execute commands via SSH: `bash ssh username@server_ip 'uname -a' ssh username@server_ip 'free -h' ssh username@server_ip 'df -h' ssh username@server_ip 'sudo ufw status' ssh username@server_ip 'systemctl status sshd'`

Interactive SSH session: `bash ssh username@server_ip pwd hostname ip addr show exit`

Workstation-to-Server Architecture: - All administration performed remotely via SSH - Server console not used directly - Demonstrates proper remote administration - Command prompts show `username@hostname` for both systems

---

Self-Assessment & Reflection

Self-Assessment Questions

Question 1: Primary distinction between Authentication and Authorization

Authentication: - Verifies identity ("Who are you?") - Confirms user is who they claim to be - Examples: passwords, biometrics, certificates - Occurs first in the AAA framework

Authorization: - Determines permissions ("What can you do?") - Defines what authenticated user can access - Examples: file permissions, role assignments - Occurs after authentication

Key Distinction: Authentication proves identity; authorization grants permissions based on that identity.

Question 2: Two common vulnerabilities mitigated by initial security setup

1. **Default Passwords**
2. Pre-set, easily guessable credentials
3. Provide direct entry points for attackers

4. Mitigation: Change all default passwords immediately

5. **Unnecessary Services**

6. Default installations include many unused services
7. Each service is a potential attack surface
8. Mitigation: Disable all non-essential services

Alternative answers: - Improper file permissions - Open network ports - Unpatched software - Weak encryption protocols

Question 3: Key difference between DAC and MAC

Discretionary Access Control (DAC): - Owner controls access permissions - Decentralized management - Higher flexibility - Lower security level - Example: Unix file permissions

Mandatory Access Control (MAC): - System enforces access based on security labels - Centralized policy management - Lower flexibility - Higher security level - Example: SELinux

Key Difference: In DAC, resource owners control access; in MAC, the system enforces access based on security policies that users cannot override.

Journal Entry Requirements Summary

Documentation Checklist

Process Management Section:

- [ ] Screenshots of `ps aux`, `top`, and `pstree` with explanations
- [ ] Examples of foreground/background process control
- [ ] Explanation of process states (R, S, D, Z, T)
- [ ] Documentation of kill vs kill -9
- [ ] Reflection on process management concepts

SSH Configuration Section:

- [ ] SSH key generation evidence
- [ ] Before and after comparison of `sshd_config` file
- [ ] Screenshots showing successful passwordless authentication
- [ ] Explanation of security improvements:
- Why ed25519 over RSA
- Benefits of PasswordAuthentication no
- Why PermitRootLogin no is important

Firewall Configuration Section:

- [ ] Complete firewall ruleset with `sudo ufw status numbered`
- [ ] Table documenting each rule and its justification
- [ ] Evidence that SSH accessible only from workstation
- [ ] Discussion of defense-in-depth strategy
- [ ] Explanation of default deny/allow policies

User Management Section:

- [ ] User creation and privilege assignment process

- [ ] List of users with sudo privileges (`getent group sudo`)
- [ ] Explanation of principle of least privilege
- [ ] Why non-root administrative users are important

Remote Administration Section:

- [ ] Multiple screenshots showing commands executed via SSH
- [ ] Evidence of workstation-to-server architecture
- [ ] Command prompts showing `username@hostname` for both systems
- [ ] Interactive session examples

Reflection Section:

- [ ] Challenges encountered and resolutions
- [ ] Security trade-offs considered
- [ ] Connection between theory (lectures) and practice (lab work)
- [ ] Lessons learned
- [ ] Areas for improvement

Technical Requirements:

- All screenshots must show visible command prompts with `username@hostname`
- Include command, output, and explanation for each task
- Update system architecture diagram to show security controls
- Commit and push to GitHub: `bash git add . git commit -m "Week 7: Initial Security Configuration" git push`

Key Takeaways

1. **Security is Multi-Layered:** The AAA framework provides comprehensive protection through authentication, authorization, and access control

2. **Initial Configuration is Critical:** Proper initial security setup prevents common vulnerabilities like default passwords and

unnecessary services

3. **Different Security Models Serve Different Needs:** DAC, MAC, and RBAC each have appropriate use cases based on security requirements
4. **Intrusion Detection is Essential:** Both HIDS and NIDS play important roles in detecting and responding to security threats
5. **Automation Enhances Security:** Bash scripting enables consistent and efficient security management
6. **Process Management is Fundamental:** Understanding process states and control mechanisms is essential for system administration
7. **SSH Key Authentication is Superior:** Key-based authentication is more secure than password authentication
8. **Firewalls Provide Network Protection:** Properly configured firewall rules restrict unauthorized access
9. **Least Privilege Principle:** Users and processes should have only the minimum permissions necessary
10. **Remote Administration Best Practices:** Secure, auditable remote access is essential for modern system administration

---

Additional Resources

Moodle Video Lectures:

- Process Management Fundamentals
- Understanding System Monitoring Tools
- Process States and Lifecycle
- Operating System Structure

Recommended Reading:

- Linux Security Documentation

- SELinux User's Guide
- SSH Protocol Specifications
- UFW Documentation

Practice Exercises:

- Configure SELinux or AppArmor
- Write bash scripts for log analysis
- Implement more complex firewall rules
- Practice incident response scenarios

End of Week 7 Journal

Table of Contents