Operating System Security Fundamentals



Initial Security Configuration

Authentication | Authorization | Access Control | Intrusion Detection | Bash Scripting

Instructor : Tanaya Bowade
& Dr Shabih Fatima

OS Security Fundamentals

The AAA Framework: Authentication, Authorization, and Access Control

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.

Authentication

Verifies the identity of a user or process. It answers the question, **"Who are you?"**

Typically involves credentials like usernames and passwords, biometrics, or digital certificates.

Authorization

Determines what an authenticated user or process is permitted to do. It answers the question, **"What can you do?"**

Involves defining permissions for specific resources or actions.

Access Control

Enforces the authorization policies. It is the mechanism that ensures the rules are followed.

Prevents unauthorized access or actions. This is where the system actively grants or denies requests based on established authorizations.

“If you were a hacker and the admin left the default password as ‘admin123’, what would you do?”

Importance of Initial Security Setup

A robust initial security configuration is paramount to prevent common vulnerabilities that attackers frequently exploit.

Common Vulnerabilities

Default Passwords

Pre-set, easily guessable passwords (e.g., "admin," "password") provide direct entry points for unauthorized access.

Unnecessary Services

Default installations include many services that are not essential, each representing a potential attack surface.

Improper Permissions

Incorrect file and directory permissions can allow unauthorized users to access sensitive system files.

Attack Vectors

Network Exploits

Default passwords or open network services can be exploited remotely, allowing attackers to gain initial system access.

Local Privilege Escalation

Improper file permissions or misconfigured services can enable a low-privileged user to gain higher-level access.

Proper initial security configuration is the first line of defense against these common vulnerabilities.

OS-Level Vulnerabilities & Impact

Understanding common vulnerabilities and their impact on system security

Common OS Vulnerabilities

Buffer Overflows -> System Crash

Writing data beyond allocated buffer size, potentially overwriting adjacent memory. Example: Heartbleed bug in OpenSSL.

Race Conditions -> Privilege Escalations

Dependencies on sequence or timing of events. Can lead to privilege escalation if events don't happen in expected order.

Improper Input Validation -> Data Leak

Failure to sanitize user input can lead to injection attacks (SQL, command injection) or buffer overflows.

Use-After-Free Vulnerabilities -> System Crash

Using memory after it has been freed can lead to data corruption, crashes, or arbitrary code execution.

System Impact

System Integrity

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

Confidentiality

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

Security Models: DAC, MAC, RBAC

Comparison of three main access control models

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

Discretionary Access Control (DAC)

Owner-centric permissions

Mandatory Access Control (MAC)

System-enforced security levels

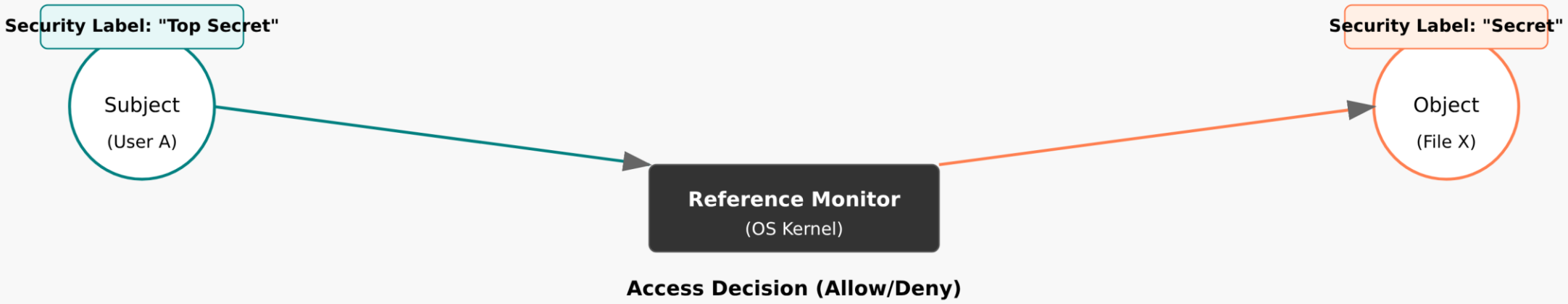
Role-Based Access Control (RBAC)

Role-privilege mapping

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

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.



Reference Monitor

The core component of the OS kernel that mediates all access attempts between subjects and objects. It enforces security policies by comparing security labels and making access decisions.

Security Labels

System-defined labels assigned by administrators that classify subjects and objects. These labels represent sensitivity levels and clearance requirements, determining what accesses are allowed.

Implementation Examples

- **SELinux:** Security-Enhanced Linux, policy-driven approach
- **AppArmor:** Path-based security, simpler than SELinux

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.
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.

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.

Host-based IDS (HIDS)

- **Monitoring Focus:** Monitors activities on a specific host or endpoint
- **Data Sources:** System logs, file system changes, process activities
- **Detection:** Internal attacks, unauthorized file access, privilege escalation

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

Network-based IDS (NIDS)

- **Monitoring Focus:** Monitors network traffic on a network segment
- **Data Sources:** Network packets, headers, and payloads in real-time
- **Detection:** Port scans, DoS attacks, unauthorized protocol usage

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

IDS Components & Workflow

Detection Techniques

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.

Advantages: High accuracy for known threats, low false positives.

Disadvantages: Ineffective against unknown or polymorphic attacks.

Anomaly-based Detection

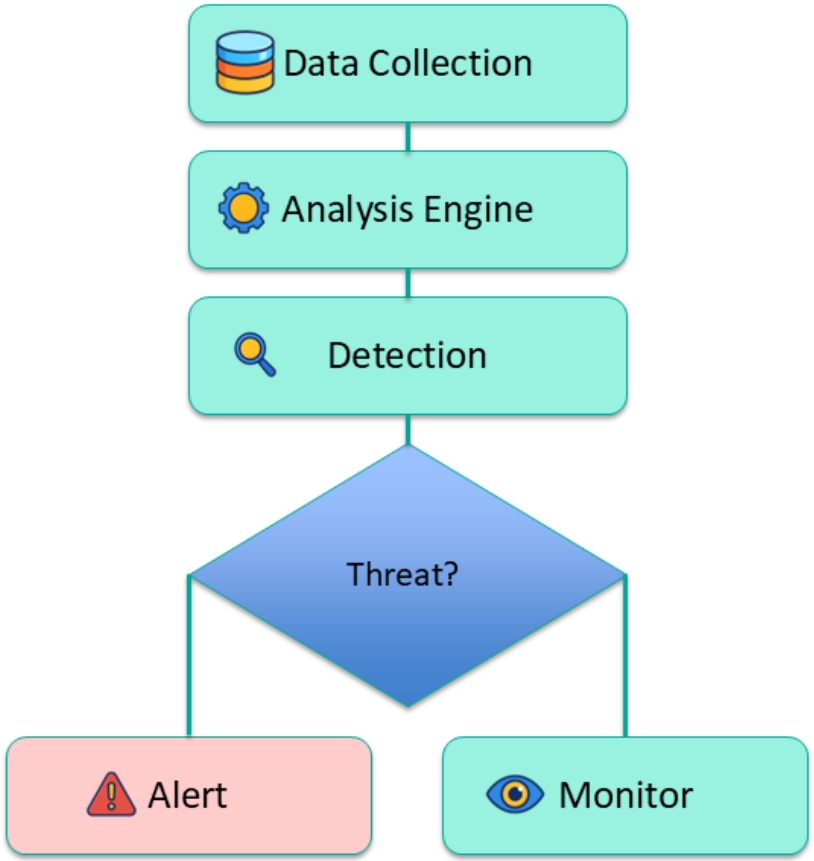
Concept: Detects deviations from established normal behavior.

How it works: Builds a baseline of normal system activity and flags deviations as suspicious.

Advantages: Can detect novel or zero-day attacks.

Disadvantages: Can generate high false positives if baseline is not accurate.

IDS Workflow



Bash Scripting for System Administration

Automating security tasks for enhanced system protection

Automation Tasks

User Auditing and Management

Automating user account management and auditing privileges

Log File Analysis

Parsing system logs to identify security events and failed logins

File Permission Control

Enforcing correct file permissions to prevent unauthorized access

Useful Commands

chmod

Change File Permissions

Example: `chmod 600 secret.txt` restricts access to the owner only

chown

Change File Ownership

Example: `chown user:group file.txt` assigns ownership

grep

Search Text Patterns

Example: `grep "failed password" /var/log/auth.log` finds failed login attempts

Bash scripting automates routine security tasks, enhancing efficiency and consistency.

Review & Self-Assessment

OS 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

Self-Assessment Questions

1. What is the primary distinction between Authentication and Authorization in the context of OS security?

2. Name two common vulnerabilities that an initial security setup aims to mitigate.

3. Explain one key difference between Discretionary Access Control (DAC) and Mandatory Access Control (MAC).