Module 3: Operating Systems Security (7 hours)

Security in Operating System Design - Module 3 Summary

1. Security in the Design of Operating Systems Operating systems play a crucial role in enforcing security. A well-designed OS must protect against unauthorized access, malware, and system failures while ensuring smooth performance.

2. Simplicity of Design

- Operating systems are inherently complex, handling multiple processes and resources.
- Security adds additional complexity, making layered design a practical approach.
- **Layered Design**: Secure OS structures follow a hierarchical model, ensuring:
  - **Separation of Privileges**: Restricts access to different layers.
  - **Encapsulation**: Prevents lower-level issues from affecting higher layers.
  - **Damage Control**: Limits the impact of security breaches.

3. Kernelized Design

- The **kernel** is the core of the OS, managing hardware and enforcing policies.
- **Security Kernel**: A specialized module responsible for enforcing access controls and security mechanisms.
- Advantages of kernelized security:
  - Centralized security enforcement.
  - Easier verification of security properties.
  - Protection against unauthorized modifications.

4. The Reference Monitor

- A concept ensuring that all access requests to system resources are controlled.
- Properties of a reference monitor (NEAT):
    - **Non-bypassable**: Every access request is checked.
    - **Evaluable**: Can be analyzed for correctness.
    - **Always Invoked**: Cannot be skipped or disabled.
    - **Tamper-proof**: Secure against modification.
- Used in access control mechanisms across OS architectures.

5. Secure Design Principles To develop a secure OS, the following principles are applied:

- **Least Privilege**: Users and applications receive only necessary permissions.
- **Economy of Mechanism**: Simple security designs reduce vulnerabilities.
- **Open Design**: Security should not rely on secrecy.
- **Complete Mediation**: Every access request must be checked.
- **Separation of Privileges**: Dividing access to minimize risks.
- **Least Common Mechanism**: Avoid sharing critical security components.
- **Ease of Use**: Security should not hinder user experience.

6. Trusted Computing Base (TCB)

- **TCB**: The collection of hardware, software, and firmware enforcing security.
- **Components**:
    - **Trusted Processes**: Secure applications executing specific tasks.
    - **Trusted Products**: Certified security software/hardware.
    - **Reference Monitor**: Enforces system-wide security policies.

    - **Trusted Path**: Ensures secure user interactions.
    - **Trusted System**: A system designed to maintain confidentiality, integrity, and availability.

7. Rootkits and Their Threats

- **Rootkits**: Malicious software designed to gain persistent access while remaining undetected.
- Types of rootkits:
    - **Kernel Rootkits**: Modify OS components to hide activities.
    - **Bootloader Rootkits**: Infect the system before OS loading.
    - **User Mode Rootkits**: Operate at the application level.
    - **Firmware Rootkits**: Attack embedded hardware components.
    - **Virtualized Rootkits**: Create hidden virtual environments for attack execution.
- **Dangers**:
    - Hard to detect and remove.
    - Provides attackers full control over infected systems.
    - Often used in cyber espionage and large-scale attacks.

8. Detecting and Removing Rootkits

- **Specialized Scanning Tools**: GMER, RootkitRevealer, chkrootkit.
- **Behavioral Analysis**: Detects suspicious system modifications.
- **Booting in Safe Mode**: Prevents rootkits from running.
- **System Reinstallation**: In severe cases, wiping and reinstalling OS is necessary.

9. Conclusion A secure OS must incorporate a well-structured design, enforce strict access controls, and protect against evolving threats like rootkits. Using a layered security model, enforcing reference monitors, and applying security best practices ensure a resilient and trustworthy computing environment.