CHAPTER 03

1. EXPLAIN IN DETAIL TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA.

The Trusted Computer System Evaluation Criteria (TCSEC) is a ***set of standards that define security requirements for computer systems.*** It was developed by the United States Department of Defence (DoD) in the 1980s, and it is still used today to evaluate the security of computer systems for government and commercial use.

The TCSEC defines ***six evaluation classes,*** from D (minimal security) to A1 (high assurance). Each class has a set of security requirements that must be met in order for the system to be evaluated at that level.

The following is a simplified explanation of the six TCSEC evaluation classes:

- **D: Minimal assurance.** Systems at this level have minimal security requirements.
- ***C1: Discretionary security protection.*** Systems at this level ***have basic security requirements,*** such as the ability to ***control access to data and resources.***
- C2: Controlled access protection. Systems at this level have **more stringent security requirements,** such as the ability to ***audit security events and verify the identity of users and processes.***
- B1: Labelled security protection. Systems at this level have even more stringent security requirements, such as the ***ability to label data with different security classifications*** and control access to data based on those classifications.
- B2: Structured protection. Systems at this level have the ***most stringent security requirements,*** such as the ability to enforce security policies through the use of a ***trusted computing base (TCB).***
- A1: Verified protection. Systems at this level have been ***evaluated and verified*** to meet the ***highest security requirements.***

The TCSEC is an important tool for evaluating the security of computer systems. It provides a set of objective standards that can be used to compare the security of different systems.

Here are some general ways to protect computer systems:

- **Physical security:** This involves measures to prevent unauthorized access to systems, such as locks, fences, and security guards.
- **Technical security:** This involves measures to protect systems from unauthorized access, use, disclosure, disruption, modification, or destruction, such as access control lists, encryption, and firewalls.
- **Administrative security:** This involves policies and procedures that define how systems should be used and protected.

2. EXPLAIN WINDOWS SECURITY COMPONENTS WITH SUITABLE DIAGRAM.

The security system components in Windows are responsible for protecting the operating system and its resources from ***unauthorized access, use, disclosure, disruption, modification, or destruction.***

The ***Windows security components*** work together to ***protect the operating system and the data stored on it*** from a variety of threats, including ***malware, unauthorized access,*** and ***data breaches.***

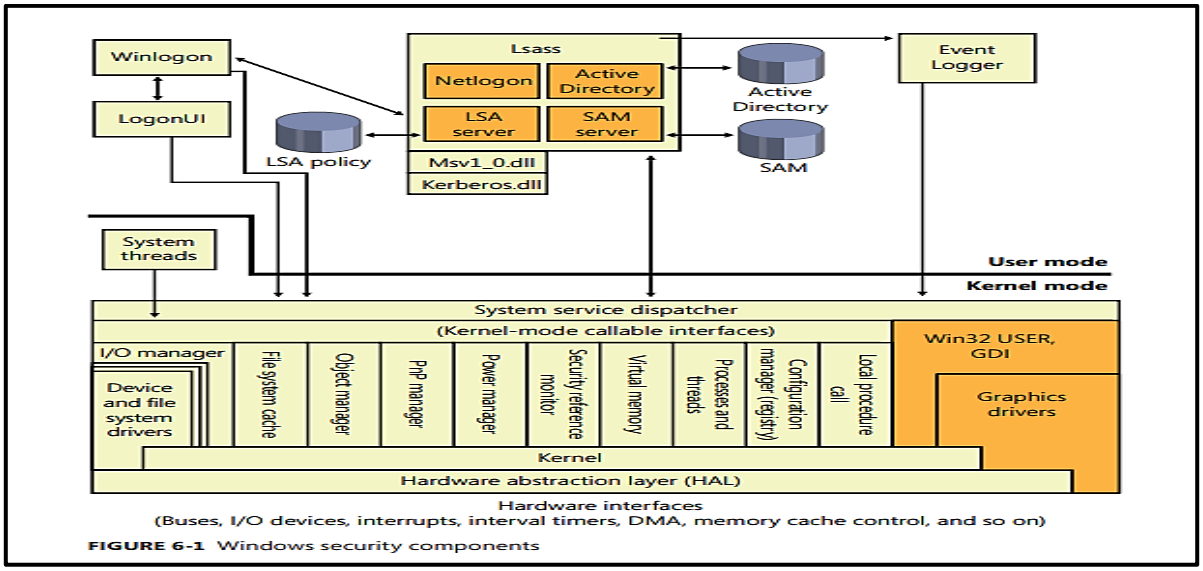


FIGURE 6-1 Windows security components

The following are some of the key components of the Windows security architecture:

1. **Security Reference Monitor (SRM):** The SRM, found in Ntoskrnl.exe, manages access rights, performs security checks on objects, handles user privileges, and creates security audit messages for system monitoring.
2. **Local Security Authority Subsystem (LSASS):** LSASS, running as Lsass.exe, oversees the local system's security policies, user authentication, and the generation of security audit messages. It loads the Local Security Authority Service (Lsasrv.dll) to carry out these functions.
3. **LSASS Policy Database:** Stored in the registry under HKLM\SECURITY, this database contains critical information such as trusted domains, access permissions, user privileges, security auditing settings, as well as secret login details used for cached domain logons and service user-account logons.
4. **Security Accounts Manager (SAM):** The SAM service, implemented as Samsrv.dll, is responsible for managing the local machine's user names, groups, and related attributes. On domain controllers, SAM stores the system's administrator recovery account definition and password.
5. **SAM Database:** Located in the registry under HKLM\SAM, the SAM database contains defined local users and groups along with their passwords and attributes, excluding domain-defined users.
6. **Active Directory:** Active Directory, implemented as Ntdsa.dll, is a directory service storing information about domain objects like users, groups, and computers. It

includes password information and privileges for domain users and groups, replicated across designated domain controllers.

7. **Authentication Packages:** Authentication packages, in the form of dynamic-link libraries, authenticate users by validating their credentials and then provide LSASS with the necessary user security identity information to generate a security token.
8. **Interactive Logon Manager (Winlogon):** Winlogon, running as Winlogon.exe, is responsible for managing interactive logon sessions and creating a user's first process upon login.
9. **Logon User Interface (LogonUI**): LogonUI, represented by LogonUI.exe, provides the user interface for system authentication and employs credential providers to collect user credentials using various methods.
10. **Credential Providers (CPs):** These includes COM objects, such as authui.dll and SmartcardCredentialProvider.dll, are used by the LogonUI process to obtain user credentials, including usernames, passwords, smartcard PINs, and biometric data.
11. **Network Logon Service (Netlogon):** Netlogon, located in Netlogon.dll, establishes secure channels with domain controllers to handle security requests for authentication and logons, including those for Active Directory.
12. **Kernel Security Device Driver (KSecDD):** KSecDD, positioned in Ksecdd.sys, facilitates secure communications between kernel mode security components, including the Encrypting File System (EFS), and LSASS in user mode.
13. **AppLocker:** AppLocker, featuring a driver (AppId.sys) and a service (AppIdSvc.dll), allows administrators to dictate which executable files, DLLs, and scripts can be utilized by specific users and groups. It helps control application access in the system.

3. EXPLAIN METHODS OF PROTECTING OBJECTS.

When it comes to protecting objects in the Windows operating system, several methods are employed to ensure the security of these objects. Here are the detailed explanations for each method:

1. **Access Checks:**
    - Access checks are performed to ascertain whether a specific user or group has the requisite permissions to execute a particular action on a secured object. This involves examining the permissions associated with the object and determining whether the user or group is allowed to perform the requested action.
2. **Security Identifiers (SIDs):**
    - Security Identifiers, or SIDs, are unique identifiers assigned to user and group accounts. These identifiers are central to the enforcement of security in Windows systems. SIDs are used in access control lists (ACLs) to specify the security context for objects, determining which users or groups are granted or denied access to the object.
3. **Virtual Service Accounts:**
    - Virtual service accounts are managed local accounts that offer an increased level of security by confining services on the network. These accounts are designed to be easily manageable and provide a higher degree of isolation, enhancing the security of services.

4. **Security Descriptors and Access Control:**
    - Security descriptors contain security-related information associated with securable objects, such as files, directories, and registry keys. They include details about the object's owner, the object's primary group, and a discretionary access control list (DACL) that specifies the users and groups that are granted or denied access to the object. Access control entries within security descriptors determine the level of access granted or denied to users or groups, governing the security of the object in question.

These methods collectively contribute to safeguarding objects within the Windows operating system, ensuring that access to sensitive data and resources is controlled and secured effectively.

How to choose the right method for protecting objects?

The best method for protecting an object depends on a number of factors, such as the ***sensitivity of the data in the object,*** the ***risk of unauthorized access,*** and the ***cost of implementing and maintaining the security*** solution.

It is also important to note that you can ***combine different methods to protect objects.*** For example, you could use ACLs to protect a file from unauthorized access and encryption to protect the file from unauthorized access if it is lost or stolen.

4. EXPLAIN THE FOLLOWING TERMS. I) SECURITY IDENTIFIERS(SIDS) II) INTEGRITY LEVELS

I) Security Identifiers (SIDs):

- **Definition:** SIDs uniquely identify entities in the Windows operating system, such as users, groups, computers, and services. A SID comprises a revision number, an identifier authority value, and a variable number of subauthority or relative identifier (RID) values.
    - **Components:**
        - **Revision Number (Example: 1):** Indicates the version of the SID structure.
        - **Identifier Authority Value (Example: 5):** Identifies the agent that issued the SID (e.g., Windows security authority).
        - Subauthority Values and RID (Example: S-1-5-21-1463437245-1224812800-863842198-1128): Represent trustees relative to the issuing authority, creating a unique SID.
    - **Generation and Uniqueness:** Windows generates SIDs with a high degree of randomness to ensure global uniqueness. Examples include domain SIDs, machine SIDs, and user/group SIDs. Predefined SIDS, such as Everyone (S-1-1-0) and Network (S-1-5-2), represent well-known groups with constant values.

II) Integrity Levels:

- **Definition:** Integrity levels are part of Windows' mandatory integrity control mechanism, assigning trust levels to processes and objects. Five primary integrity levels are defined: Untrusted, Low, Medium, High, and System.

- *System:* The highest integrity level. System processes and objects have the highest level of trust.
- *High:* A high integrity level. Processes and objects with a high integrity level are considered to be trusted.
- *Medium:* The *default integrity level.* Most processes and objects are assigned a medium integrity level.
- *Low:* The lowest integrity level. Processes and objects with a *low integrity level are considered to be untrusted.*

- **Representation:** Integrity levels are represented by SIDs (e.g., S-1-16-0x0 for Untrusted, S-1-16-0x1000 for Low).
- **Usage:** Integrity levels control access, preventing unauthorized interactions between processes and objects. They define the trust required to access an object or execute code.
- **Inheritance and Creation:** Processes typically inherit the integrity level of their parent, while objects have an implicit integrity level if not specified during creation. Explicit integrity levels can be assigned to objects to prevent lower integrity processes from accessing or modifying them.
- **Security Policies:** Mandatory policies associated with integrity levels, like No-Write-Up, restrict write access from lower integrity processes. Other policies include No-Read-Up and No-Execute-Up, providing protection against information leakage and code execution.

In summary, Security Identifiers (SIDs) uniquely identify entities in Windows using a structured format, while Integrity Levels, represented by SIDs, ensure a controlled trust environment and prevent unauthorized access and interactions based on predefined policies.

Example

The following example shows how SIDs and integrity levels can be used to control access to a file:

File: C:\Windows\System32\ntoskrnl.exe
Owner SID: S-1-5-18
Integrity Level: System

Only processes with a System integrity level and the appropriate permissions will be able to access the ***ntoskrnl.exe*** file.

5. WHAT IS IMPERSONATION? EXPLAIN IN BRIEF

Impersonation is a crucial feature in Windows' security model, commonly used in client/server programming. It simplifies server tasks, allowing it to temporarily adopt the security profile of a client making a resource request. This is particularly useful when a server needs to grant access to resources like files, printers, or databases.

Key Points:

1. **Purpose:**
   - Servers, when requested by clients, can impersonate the client's security context to perform actions on their behalf, accessing resources on the server.
   - Impersonation simplifies security-related tasks, such as determining if a user has the necessary permissions to perform operations on a resource.

2. **How it Works:**
   - The server notifies the Security Reference Monitor (SRM) that it wants to impersonate a client.
   - The server temporarily adopts the security profile of the client, enabling it to access resources based on the client's permissions.
   - Access validations are carried out by the SRM using the impersonated client's security context.

3. **Mechanisms:**
   - Various Windows APIs support impersonation, including ImpersonateNamedPipeClient, DdeImpersonateClient, RpcImpersonateClient, ImpersonateSelf, and ImpersonateSecurityContext.

4. **Limitations:**
   - Impersonation is performed within a specific thread.
   - It's convenient for specific actions but cannot execute an entire program in the context of a client.
   - Accessing network resources may require delegation-level impersonation and sufficient credentials.

5. **Reverting:**
   - After the server completes the task, it reverts to its primary security context.

6. **Ensuring Security:**
   - Impersonation requires client consent, and clients can specify the level of impersonation a server is allowed to perform using Security Quality of Service (SQOS) flags.
   - SQOS flags include SecurityAnonymous, SecurityIdentification, SecurityImpersonation, and SecurityDelegation, each allowing different levels of operations in the client's security context.

7. **Preventing Spoofing:**
   - To prevent security risks, a thread cannot impersonate a token of higher integrity than its own, ensuring the integrity of the system.

In essence, impersonation facilitates secure and controlled access to resources on behalf of clients, enhancing the efficiency of server applications in a Windows environment.

6. ANALYSE ACCOUNT RIGHTS AND PRIVILEGES

Account rights and privileges play a crucial role in defining the capabilities and permissions of users and processes in the Windows operating system. Here's a brief analysis:

Account Rights:
Account rights are enforced by the Local Security Authority (LSA) and are applied system-wide. They include:

- Examples of Account Rights:
  - SeAssignPrimaryTokenPrivilege
  - SeAuditPrivilege
  - SeBackupPrivilege
  - SeChangeNotifyPrivilege
  - SeShutdownPrivilege
- Enforcement:
  - Enforced by the LSA, providing a centralized control mechanism.
  - Checked by various system components based on specific operations.
- Characteristics:
  - Applied system-wide.
  - Managed centrally by the LSA.
  - Provide fundamental system-level permissions.

Privileges:
Privileges are more granular than account rights and are enforced by various components based on their specific functionalities. They include:

- Examples of Privileges:
  - SeDebugPrivilege
  - SeTakeOwnershipPrivilege
  - SeRestorePrivilege
  - SeLoadDriverPrivilege
  - SeCreateTokenPrivilege
- Enforcement:
  - Enforced by specific components like the process manager, NTFS, or the Security Accounts Manager.
  - Checked using APIs like PrivilegeCheck or LsaEnumerateAccountRights in user mode and SeSinglePrivilegeCheck or SePrivilegeCheck in kernel mode.
- Characteristics:
  - Can be enabled or disabled dynamically.
  - Checked for and enforced by components relevant to their usage.
  - Provide fine-grained control over specific actions.

Super Privileges:
Some privileges are exceptionally powerful, essentially granting a user "superuser" capabilities. These include Debug programs, Take ownership, Restore files and directories, Load and unload device drivers, Create a token object, and Act as part of operating system.

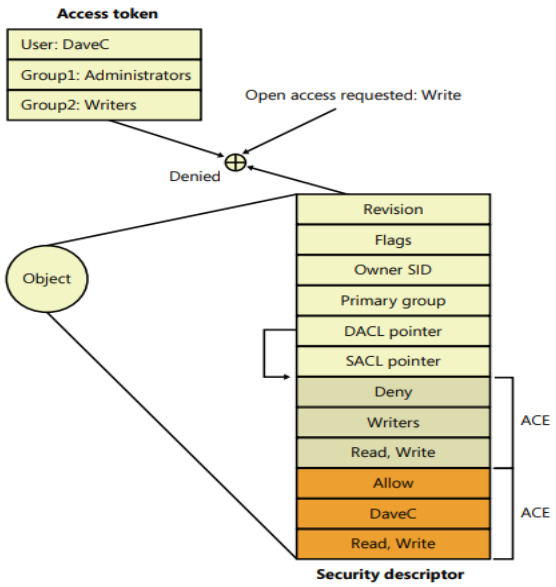
- Potential Exploitation:
  - Users with these privileges can execute code that grants additional privileges or performs unauthorized operations.

- For example, Debug programs can be exploited to inject code into other processes, elevating privileges.

In summary, account rights and privileges work together to define the scope and capabilities of users and processes in the Windows operating system. While account rights are system-wide and centrally managed, privileges offer more fine-grained control and are enforced by specific components. Super privileges, when misused, can lead to significant security risks and unauthorized system actions.

7. LIST AND EXPLAIN THE ATTRIBUTES OF SECURITY DESCRIPTOR.

The security descriptor in Windows OS is a ***data structure*** that ***contains information about the security of an object.*** It is associated with every object in Windows OS, such as files, folders, and processes.



Here is a brief explanation of each attribute of the security descriptor:

- **Owner:** The owner of an object is the user or group that has the most control over the object. The owner can grant and deny permissions to other users and groups.
- **Revision number**: Revision number is a version of the ***SRM security model*** used to create the descriptor.
- **Flags**: Optional modifiers that ***define the behaviour or characteristics*** of the descriptor.
- **Group:** The primary group is the group that the object is associated with. The members of the primary group have the same permissions to the object as the owner of the object.

- **DACL:** The DACL specifies the permissions that users and groups have to access an object. The DACL can contain permissions for both individual users and groups.

- **SACL:** The SACL specifies the types of access attempts that generate audit records for the object. The SACL can contain permissions for both individual users and groups.

The security descriptor is used by the Windows OS to ***determine who has access to an object and what actions they can perform on the object.*** For example, when a user tries to open a file, the Windows OS will check the security descriptor of the file to determine whether the user has permission to open the file.

6. DESCRIBE THE PROCESS TOKEN IN WINDOWS OPERATING SYSTEM / LIST AND EXPLAIN VARIOUS FIELDS OF PROCESS TOKEN.

Introduction

A process token in Windows is an object that contains information about the ***security context of a process.*** This information includes the ***user account that the process is running under, the privileges that the process has,*** and the ***groups that the user belongs to.***

The process token is used by the operating system to determine ***what resources the process is allowed to access and what actions it is allowed to perform.*** For example, if a process tries to open a file, the operating system will use the process token to determine whether the process has the necessary permissions to open the file.

Process tokens are created when a process is created and are ***destroyed when the process is terminated.*** The ***process token is associated with the process throughout its lifetime.***

The following are some of the key fields in a process token:

- **Token Source:** The SID of the user or group that created the process.
- **Token Type:** The type of token. The most common type of token is ***Token Primary.***
- **TokenIsElevated:** A ***Boolean value indicating whether the token is elevated.*** An elevated token has all privileges, including administrative privileges.
- **TokenImpersonationLevel:** The impersonation level of the token. The impersonation level determines what resources the process can access when impersonating another user.
- **TokenSessionId:** The SID of the session that the process is associated with.
- **Token Groups:** A list of the groups that the user is a member of.
- **Token Privileges:** A list of the privileges that the process has.
- **Token Owner:** The SID of the owner of the token.
- **TokenPrimaryGroup**: The SID of the primary group of the user.
- **TokenDefaultDacl:** The default access control list (DACL) for the token. The DACL determines what permissions users and groups have to access the processes' resources.
- **Token User:** The SID of the user associated with the token.

Uses of process tokens

Process tokens can be used to control the ***access of processes to resources*** and to ***perform other security-related tasks.***

For example, you can use a process token to:

- Determine whether a process has the necessary permissions to access a resource.
- Impersonate another user.
- Elevate a process to have administrative privileges.

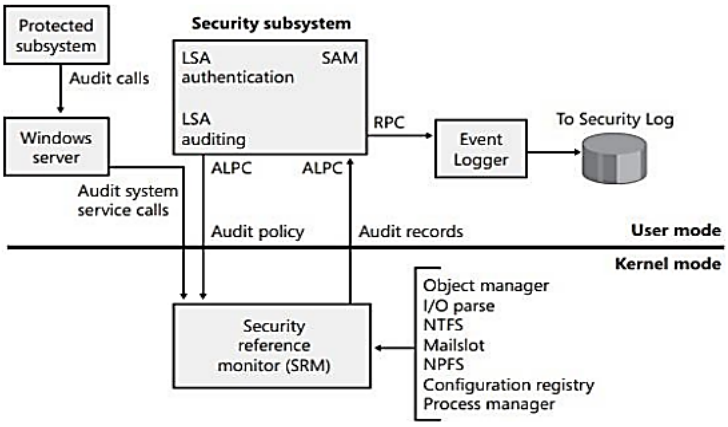
Process tokens are an important part of the Windows security architecture. By understanding how process tokens work, you can better control what processes can do and what resources they can access.

SECURITY AUDITING:

Security auditing in Windows OS is like keeping a ***detailed record of important security-related events*** that happen on your computer. This helps to ***identify and investigate any potential security issues*** and make your system more secure.

Here's a simpler breakdown:

1. **What is Security Auditing?**
    - Security auditing is like taking notes on important security events that occur on your Windows computer. These notes are helpful for spotting and understanding security problems and making your computer more secure.



2. **How is it Done?**
    - The Security Reference Monitor (SRM) is like a security guard for your computer. It keeps track of security rules and checks if actions meet these rules. When something important happens (like someone trying to log in or access a file), the SRM writes it down as a note.
3. **Components Involved:**
    - ***The Security Reference Monitor (SRM)*** is like the security guard who keeps track of events.
    - ***The Local Security Authority (LSA)*** helps organize the notes.
    - The ***Event Logger*** is like the logbook where all the notes are written down.

4. **Why is it Useful?**
    - **Keeping these detailed notes is helpful because:**
        - It helps you quickly respond to and understand security issues.
        - Many rules and laws require organizations to keep these notes to protect important information.
        - It's like having a security camera for your computer that watches for any unauthorized access to important data.

So, security auditing is like maintaining a record of what's happening on your computer, helping you keep it safe and secure.