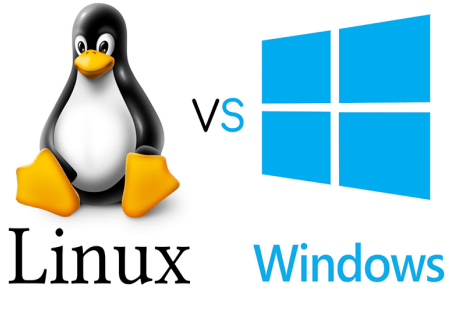
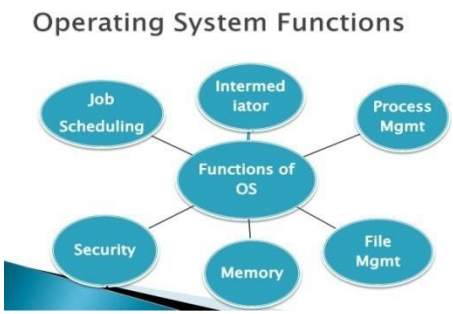
CIT210 OP. SYS. MANAG.

CSUN
COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Week13Day2: Windows Security, TCSEC, Common Criteria, SRM, Protecting Objects, Security Identifiers, Integrity Levels, Tokens, Access Rights and Privileges, Bit Locker Drive Encryption





Trusted Computer System Evaluation Criteria

- The TCSEC standard consists of “levels of trust” ratings, where higher levels build on lower levels by adding more rigorous protection and validation requirements.
- No operating system meets the A1, or “Verified Design,” rating.
- Although a few operating systems have earned one of the B-level ratings, C2 is considered sufficient and the highest rating practical for a general-purpose operating system.

TCSEC Rating Levels

Rating	Description
A1	Verified Design
B3	Security Domains
B2	Structured Protection
B1	Labeled Security Protection
C2	Controlled Access Protection
C1	Discretionary Access Protection (obsolete)
D	Minimal Protection

Windows TCSEC Rating

- In July 1995, Windows NT 3.5 (Workstation and Server) with Service Pack 3 was the first version of Windows NT to earn the C2 rating.
- In March 1999, Windows NT 4 with Service Pack 3 achieved an E3 rating from the U.K. government's Information Technology Security (ITSEC) organization, a rating equivalent to a U.S. C2 rating.
- In November 1999, Windows NT 4 with Service Pack 6a earned a C2 rating in both stand-alone and networked configurations.

Windows TCSEC Rating

- The following were the key requirements for a C2 security rating, and they are still considered the core requirements for any secure operating system:
  - A secure logon facility, which requires that users can be uniquely identified and that they must be granted access to the computer only after they have been authenticated in some way.
  - Discretionary access control, which allows the owner of a resource (such as a file) to determine who can access the resource and what they can do with it.
  - The owner grants rights that permit various kinds of access to a user or to a group of users.

Windows TCSEC Rating

- The following were the key requirements for a C2 security rating, and they are still considered the core requirements for any secure operating system:
  - Security auditing, which affords the ability to detect and record security-related events or any attempts to create, access, or delete system resources.
  - Logon identifiers record the identities of all users, making it easy to trace anyone who performs an unauthorized action.

Windows TCSEC Rating

- **The following were the key requirements for a C2 security rating, and they are still considered the core requirements for any secure operating system:**
  - **Object reuse protection, which prevents users from seeing data that another user has deleted or from accessing memory that another user previously used and then released.**
  - **For example, in some operating systems, it's possible to create a new file of a certain length and then examine the contents of the file to see data that happens to have occupied the location on the disk where the file is allocated. This data might be sensitive information that was stored in another user's file but had been deleted. Object reuse protection prevents this potential security hole by initializing all objects, including files and memory, before they are allocated to a user.**

Windows Security System Components

- **Security reference monitor (SRM)**
  - **A component in the Windows executive**
  - **(%SystemRoot%\System32\Ntoskrnl.exe) that is responsible for defining the access token data structure to represent a security context, performing security access checks on objects, manipulating privileges (user rights), and generating any resulting security audit messages.**

Windows Security System Components

- **Local Security Authority subsystem (LSASS)**
  - A user-mode process running the image
  - %SystemRoot%\System32\Lsass.exe
  - that is responsible for the local system security policy (such as which users are allowed to log on to the machine, password policies, privileges granted to users and groups, and the system security auditing settings), user authentication, and sending security audit messages to the Event Log.
  - The Local Security Authority service
  - (Lsasrv—%SystemRoot%\System32\Lsasrv.dll), a library that LSASS loads, implements most of this functionality.

Windows Security System Components

- **LSASS policy database**
  - A database that contains the local system security policy settings.
  - This database is stored in the registry in an ACL-protected area under HKLM\SECURITY.
  - It includes such information as what domains are entrusted to authenticate logon attempts, who has permission to access the system and how (interactive, network, and service logons), who is assigned which privileges, and what kind of security auditing is to be performed.
  - The LSASS policy database also stores “secrets” that include logon information used for cached domain logons and Windows service user-account logons.

Windows Security System Components

- **Security Accounts Manager (SAM)**
  - A service responsible for managing the database that contains the user names and groups defined on the local machine.
  - The SAM service, which is implemented as %SystemRoot%\System32\Samsrv.dll, is loaded into the LSASS process.

Windows Security System Components

- **SAM database**
  - A database that contains the defined local users and groups, along with their passwords and other attributes.
  - On domain controllers, the SAM does not store the domain defined users, but stores the system's administrator recovery account definition and password.

This database is stored in the registry under HKLM\SAM.

Windows Security System Components

- **Active Directory**
  - A directory service that contains a database that stores information about objects in a domain.
  - A *domain is a collection of computers and their associated security* groups that are managed as a single entity.
  - Active Directory stores information about the objects in the domain, including users, groups, and computers.
  - Password information and privileges for domain users and groups are stored in Active Directory, which is replicated across the computers that are designated as domain controllers of the domain. The Active Directory server, implemented as %SystemRoot%\System32\Ntdsa.dll, runs in the LSASS process.

Windows Security System Components

- **Authentication packages**
  - These include dynamic-link libraries (DLLs) that run both in the context of the LSASS process and client processes, and implement Windows authentication policy.
  - An authentication DLL is responsible for authenticating a user, by checking whether a given user name and password match, and if so, returning to the LSASS information detailing the user's security identity, which LSASS uses to generate a token.

Windows Security System Components

- **Interactive logon manager (Winlogon)**
  - **A user-mode process running**
  - **%SystemRoot% \System32\Winlogon.exe that is responsible for responding to the (Secure Attention Sequence) SAS and for managing interactive logon sessions. Winlogon creates a user's first process when the user logs on.**

Windows Security System Components

- Logon user interface (LogonUI)
  - A user-mode process running
  - %SystemRoot%\System32\LogonUI.exe that presents users with the user interface they can use to authenticate themselves on the system.
  - LogonUI uses credential providers to query user credentials through various methods.

Windows Security System Components

- **Credential providers (CPs)**
  - **In-process COM objects that run in the LogonUI process (started**
  - **on demand by Winlogon when the SAS is performed) and used to obtain a user's name and password, smartcard PIN, or biometric data (such as a fingerprint).**
  - **The standard CPs are %SystemRoot%\System32\authui.dll and %SystemRoot%\System32 \SmartcardCredentialProvider.dll.**

Windows Security System Components

- **Network logon service (Netlogon)**
  - A Windows service (%SystemRoot%\System32 \Netlogon.dll) that sets up the secure channel to a domain controller, over which security requests—such as an interactive logon (if the domain controller is running Windows NT 4) or LAN Manager and NT LAN Manager (v1 and v2) authentication validation—are sent.
  - Netlogon is also used for Active Directory logons.

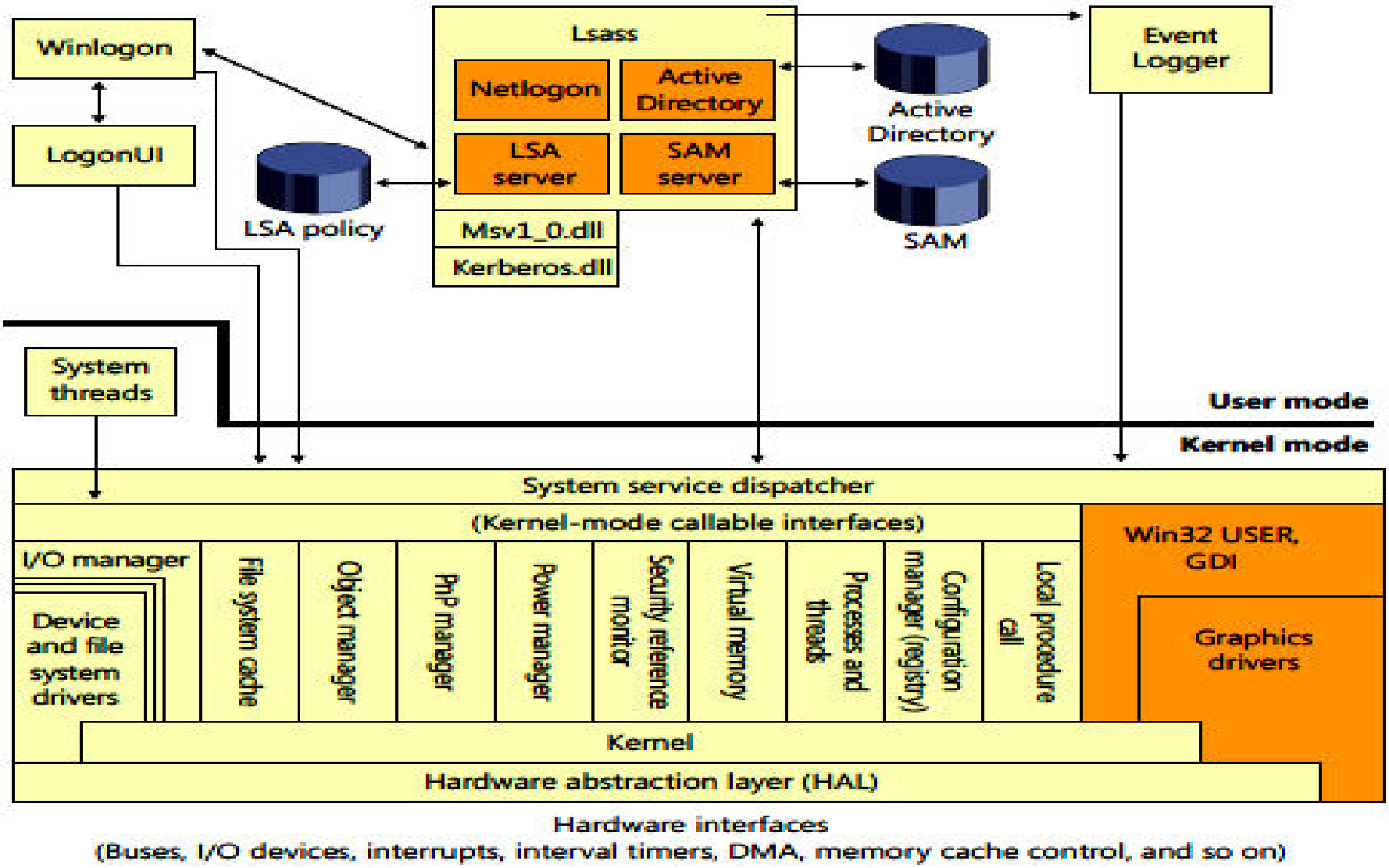
Windows Security System Components

- **Kernel Security Device Driver (KSecDD)**
  - **A kernel-mode library of functions that implement the advanced local procedure call (ALPC) interfaces that other kernel mode security components, including the Encrypting File System (EFS), use to communicate with LSASS in user mode. KSecDD is located in %SystemRoot%\System32\Drivers\Ksecdd.sys.**

Windows Security System Components

- **AppLocker**
  - **A mechanism that allows administrators to specify which executable files, DLLs, and scripts can be used by specified users and groups. AppLocker consists of a driver**
  - **(%SystemRoot%\System32\Drivers\AppId.sys) and a service (%SystemRoot%\System32\AppIdSvc.dll) running in a SvcHost process.**

Windows Security Components

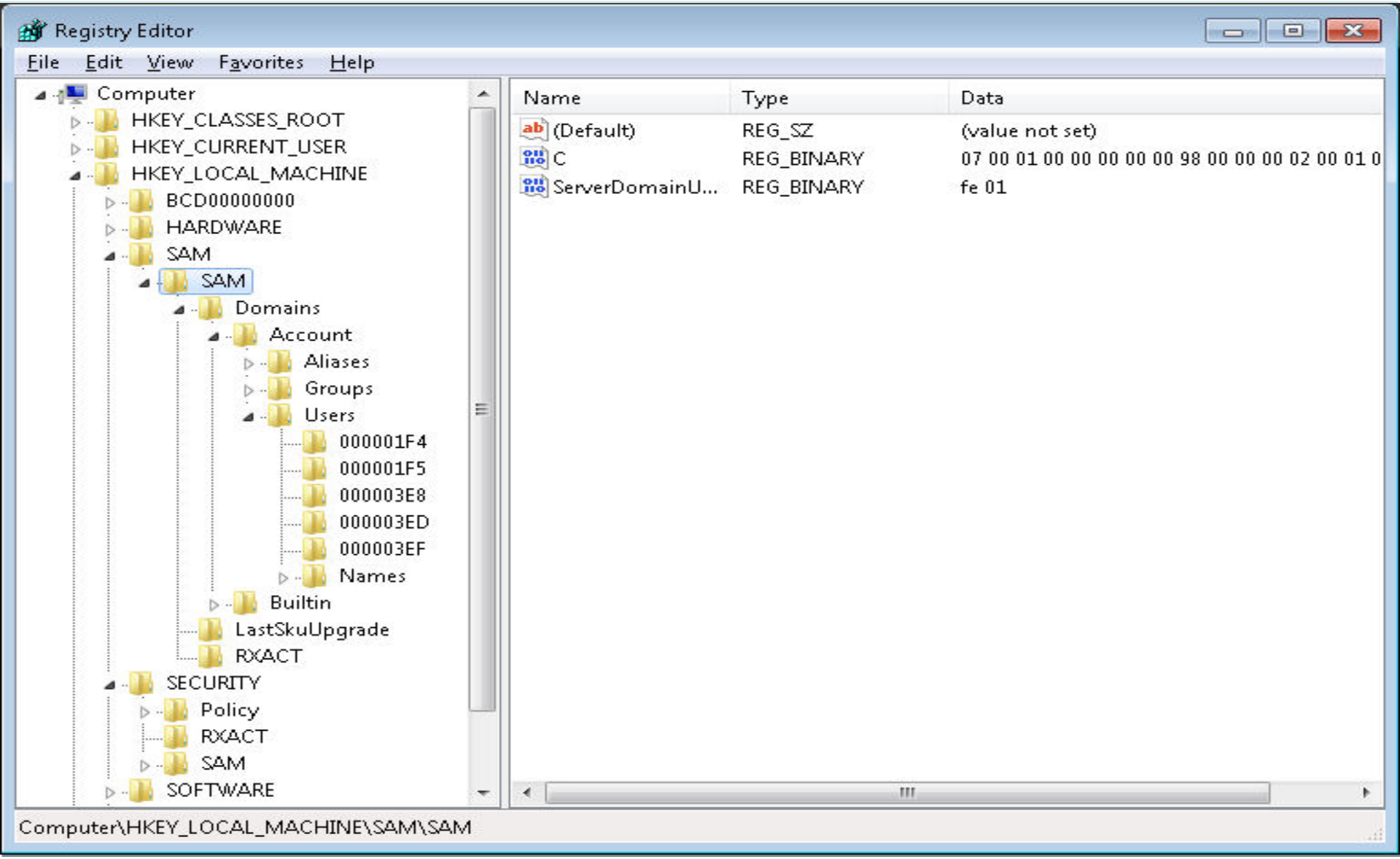


Looking Inside HKLM\SAM and HKLM\Security

- The security descriptors associated with the SAM and Security keys in the registry prevent access by any account other than the local system account.
- One way to gain access to these keys for exploration is to reset their security, but that can weaken the system's security.
- Another way is to execute Regedit.exe while running as the local system account.
- This can be done using the PsExec tool from Windows Sysinternals with the *–s option, as shown here:*

- C:\>psexec –s –i –d c:\windows\regedit.exe

Looking Inside HKLM\SAM and HKLM\Security

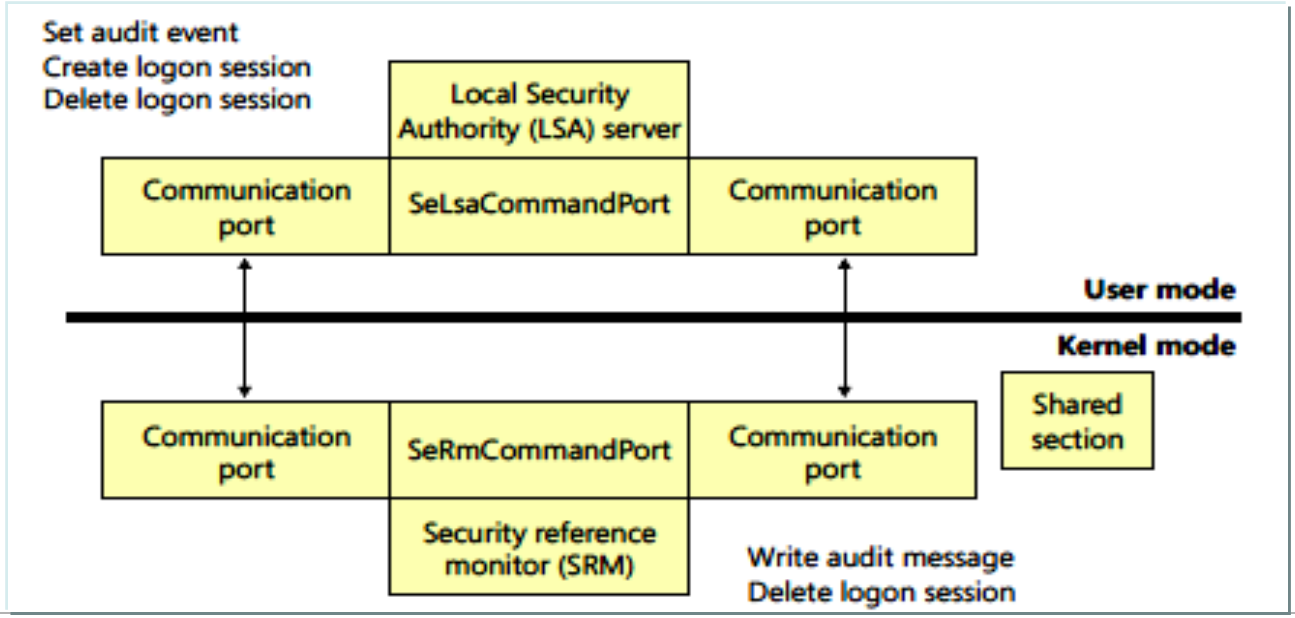


Communication paths after system initialization

- The SRM, which runs in kernel mode, and LSASS, which runs in user mode, communicate using the ALPC facility.
- During system initialization, the SRM creates a port, named SeRmCommandPort, to which LSASS connects.
- When the LSASS process starts, it creates an ALPC port named SeLsaCommandPort.
- The SRM connects to this port, resulting in the creation of private communication ports.
- The SRM creates a shared memory section for messages longer than 256 bytes, passing a handle in the connect call.
-

Communication paths after system initialization

- Once the SRM and LSASS connect to each other during system initialization, they no longer listen on their respective connect ports.
- Therefore, a later user process has no way to connect successfully to either of these ports for malicious purposes— the connect request will never complete.



Protecting Objects

- Object protection and access logging is the essence of discretionary access control and auditing.
- The objects that can be protected on Windows include files, devices, mailslots, pipes (named and anonymous), jobs, processes, threads, events, keyed events, event pairs, mutexes, semaphores, shared memory sections, I/O completion ports, LPC ports, waitable timers, access tokens, volumes, window stations, desktops, network shares, services, registry keys, printers, Active Directory objects, and so on—theoretically, anything managed by the executive object manager.

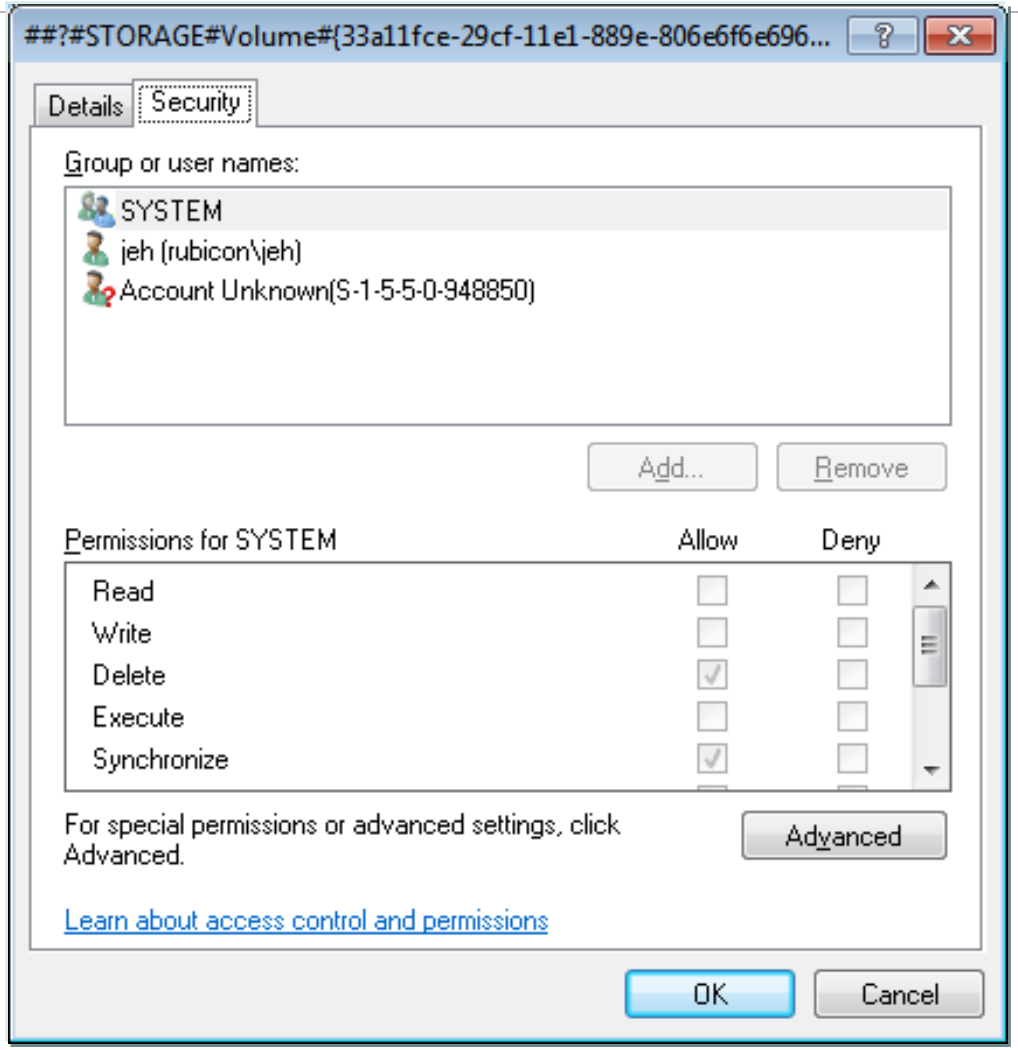
Protecting Objects

- In practice, objects that are not exposed to user mode (such as driver objects) are usually not protected.
- Kernel-mode code is trusted and usually uses interfaces to the object manager that do not perform access checking.
- Because system resources that are exported to user mode (and hence require security validation) are implemented as objects in kernel mode, the Windows object manager plays a key role in enforcing object security.

Protecting Objects

- To control who can manipulate an object, the security system must first be sure of each user's identity.
- This need to guarantee the user's identity is the reason that Windows requires authenticated logon before accessing any system resources.
- When a process requests a handle to an object, the object manager and the security system use the caller's security identification and the object's security descriptor to determine whether the caller should be assigned a handle that grants the process access to the object it desires.

Protecting Objects



An executive object and its security descriptor, viewed by Winobj

Protecting Objects

- A thread can assume a different security context than that of its process.
- This mechanism is called impersonation, and when a thread is impersonating, security validation mechanisms use the thread's security context instead of that of the thread's process.
- When a thread isn't impersonating, security validation falls back on using the security context of the thread's owning process.
- It's important to keep in mind that all the threads in a process share the same handle table, so when a thread opens an object—even if it's impersonating—all the threads of the process have access to the object.

Protecting Objects

- Sometimes, validating the identity of a user isn't enough for the system to grant access to resource that should be accessible by the account.
- Logically, one can think of a clear distinction between a service running under the Alice account and an unknown application that Alice downloaded while browsing the Internet.
- Windows achieves this kind of intra-user isolation with the Windows integrity mechanism, which implements integrity levels.
- The Windows integrity mechanism is used by User Account Control (UAC) elevations, Protected Mode Internet Explorer (PMIE), and User Interface Privilege Isolation (UIPI).

Access Checks

- The Windows security model requires that a thread specify up front, at the time that it opens an object, what types of actions it wants to perform on the object.
- The object manager calls the SRM to perform access checks based on a thread's desired access, and if the access is granted, a handle is assigned to the thread's process with which the thread (or other threads in the process) can perform further operations on the object.
- The object manager records the access permissions granted for a handle in the process' handle table.

Security Identifiers

- Instead of using names (which might or might not be unique) to identify entities that perform actions in a system, Windows uses security identifiers (SIDs).
- Users have SIDs, and so do local and domain groups, local computers, domains, domain members, and services.
- A SID is a variable-length numeric value that consists of a SID structure revision number, a 48-bit identifier authority value, and a variable number of 32-bit sub authority or relative identifier (RID) values.

Security Identifiers

- The authority value identifies the agent that issued the SID, and this agent is typically a Windows local system or a domain.
- Subauthority values identify trustees relative to the issuing authority, and RIDs are simply a way for Windows to create unique SIDs based on a common base SID.
- Because SIDs are long and Windows takes care to generate truly random values within each SID, it is virtually impossible for Windows to issue the same SID twice on machines or domains anywhere in the world.

Security Identifiers

- When displayed textually, each SID carries an S prefix, and its various components are separated with hyphens:
- *S-1-5-21-1463437245-1224812800-863842198-1128*
- In this SID, the revision number is 1, the identifier authority value is 5 (the Windows security authority), and four subauthority values plus one RID (1128) make up the remainder of the SID.
- This SID is a domain SID, but a local computer on the domain would have a SID with the same revision number, identifier authority value, and number of subauthority values.

Security Identifiers

- When you install Windows, the Windows Setup program issues the computer a machine SID.
- Windows assigns SIDs to local accounts on the computer.
- Each local-account SID is based on the source computer's SID and has a RID at the end.
- RIDs for user accounts and groups start at 1000 and increase in increments of 1 for each new user or group.

Security Identifiers

- Similarly, Dcpromo.exe (Domain Controller Promote), the utility used to create a new Windows domain, reuses the computer SID of the computer being promoted to domain controller as the domain SID, and it re-creates a new SID for the computer if it is ever demoted.
- Windows issues to new domain accounts SIDs that are based on the domain SID and have an appended RID (again starting at 1000 and increasing in increments of 1 for each new user or group). A RID of 1028 indicates that the SID is the twenty-ninth SID the domain issued.

Security Identifiers

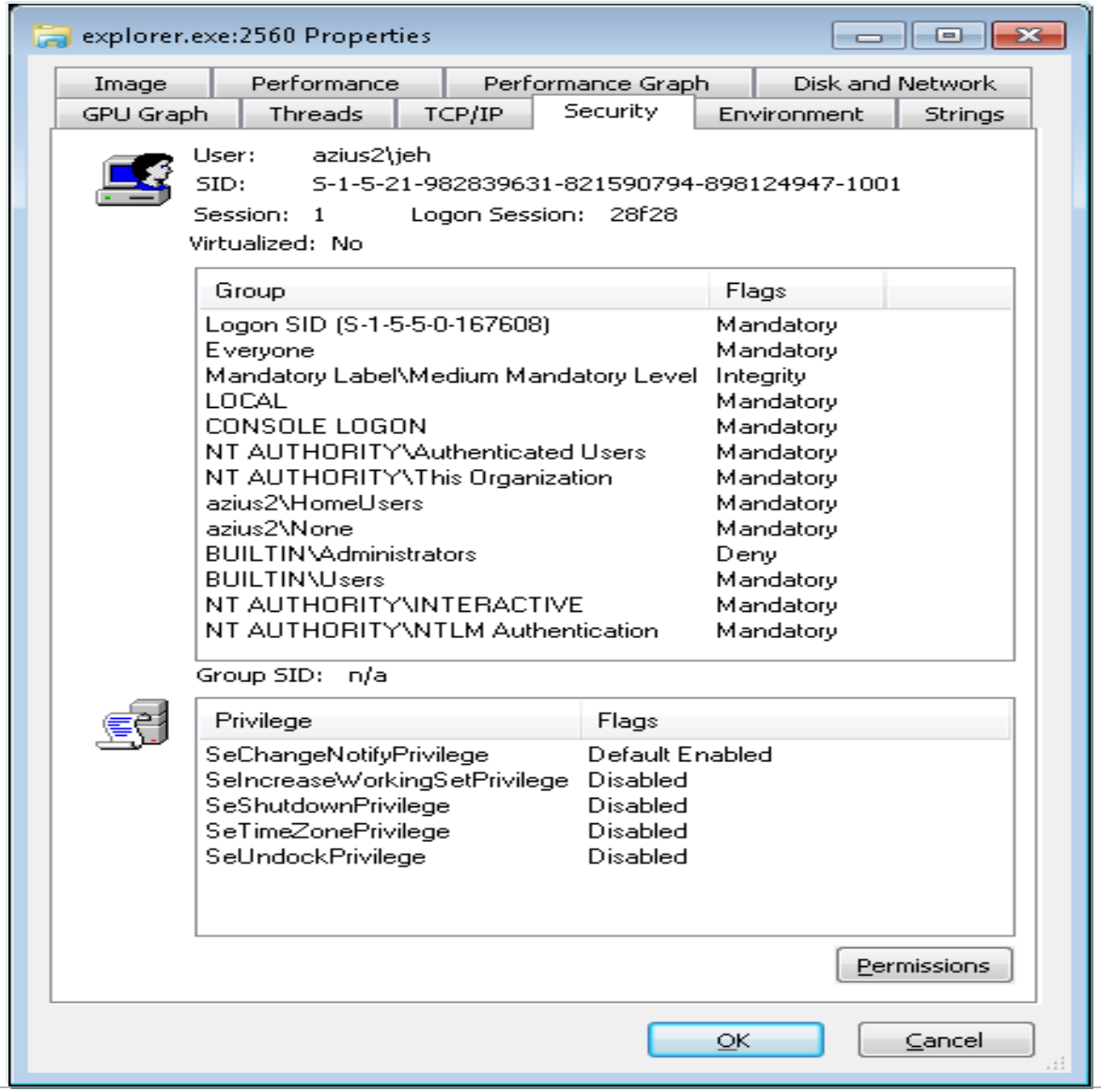
- Windows issues SIDs that consist of a computer or domain SID with a predefined RID to many predefined accounts and groups.
- For example, the RID for the administrator account is 500, and the RID for the guest account is 501.
- A computer's local administrator account, for example, has the computer SID as its base with the RID of 500 appended to it:
- *S-1-5-21-13124455-12541255-61235125-500*

Security Identifiers

TABLE 6-2 A Few Well-Known SIDs

SID	Group	Use
S-1-0-0	Nobody	Used when the SID is unknown.
S-1-1-0	Everyone	A group that includes all users except anonymous users.
S-1-2-0	Local	Users who log on to terminals locally (physically) connected to the system.
S-1-3-0	Creator Owner ID	A security identifier to be replaced by the security identifier of the user who created a new object. This SID is used in inheritable ACEs.
S-1-3-1	Creator Group ID	Identifies a security identifier to be replaced by the primary-group SID of the user who created a new object. Use this SID in inheritable ACEs.
S-1-9-0	Resource Manager	Used by third-party applications performing their own security on internal data (such as Microsoft Exchange).

Security Identifiers



Integrity Levels

- Integrity levels can override discretionary access to differentiate a process and objects running as and owned by the same user, offering the ability to isolate code and data within a user account.
- The mechanism of mandatory integrity control (MIC) allows the SRM to have more detailed information about the nature of the caller by associating it with an integrity level.
- It also provides information on the trust required to access the object by defining an integrity level for it.

Integrity Levels

- **These integrity levels are specified by a SID. Though integrity levels can be arbitrary values, the system uses five primary levels to separate privilege levels, as described in Table**

SID	Name (Level)	Use
S-1-16-0x0	Untrusted (0)	Used by processes started by the Anonymous group. It blocks most write access.
S-1-16-0x1000	Low (1)	Used by Protected Mode Internet Explorer. It blocks write access to most objects (such as files and registry keys) on the system.
S-1-16-0x2000	Medium (2)	Used by normal applications being launched while UAC is enabled.
S-1-16-0x3000	High (3)	Used by administrative applications launched through elevation when UAC is enabled, or normal applications if UAC is disabled and the user is an administrator.
S-1-16-0x4000	System (4)	Used by services and other system-level applications (such as Wininit, Winlogon, Smss, and so forth).

Integrity Levels

- Every process has an integrity level that is represented in the process' token and propagated according to the following rules:
  - A process normally inherits the integrity level of its parent (which means an elevated command prompt will spawn other elevated processes).
  - If the file object for the executable image to which the child process belongs has an integrity level and the parent process' integrity level is medium or higher, the child process will inherit the lower of the two.
  - A parent process can create a child process with an explicit integrity level lower than its own

Integrity Levels

- Integrity levels can override discretionary access to differentiate a process and objects running as and owned by the same user, offering the ability to isolate code and data within a user account.
- The mechanism of mandatory integrity control (MIC) allows the SRM to have more detailed information about the nature of the caller by associating it with an integrity level.
- It also provides information on the trust required to access the object by defining an integrity level for it.

Tokens

- The SRM uses an object called a token (or access token) to identify the security context of a process or thread.
- A security context consists of information that describes the account, groups, and privileges associated with the process or thread.
- Tokens also include information such as the session ID, the integrity level, and UAC virtualization state.
- During the logon process LSASS creates an initial token to represent the user logging on. It then determines whether the user logging on is a member of a powerful group or possesses a powerful privilege.

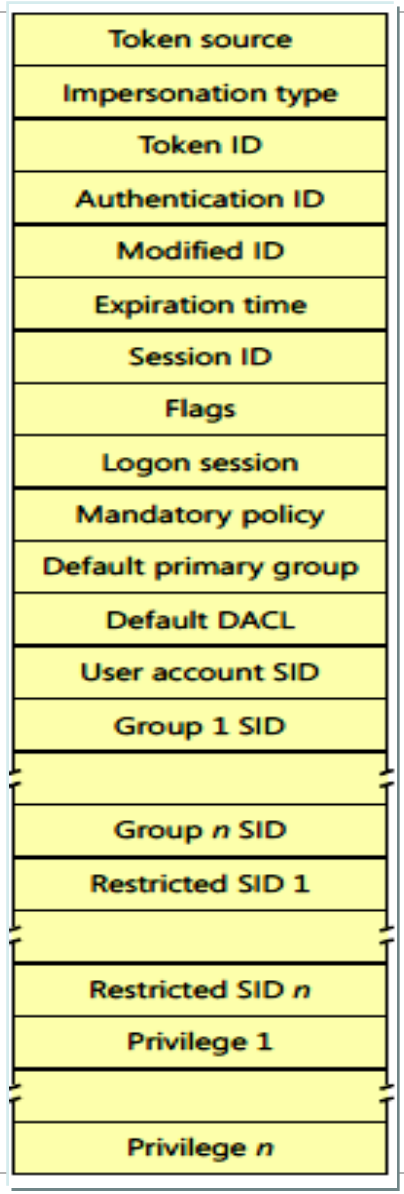
Tokens

- If one or more of these groups or privileges are present, LSASS creates a restricted token for the user (also called a filtered admin token), and it creates a logon session for both.
- The standard user token is attached to the initial process or processes that Winlogon starts (by default, Userinit.exe).

Tokens

- Because child processes by default inherit a copy of the token of their creators, all processes in the user's session run under the same token.
- You can also generate a token by using the Windows *LogonUser* function.
- You can then use this token to create a process that runs within the security context of the user logged on through the *LogonUser function by passing the token to the Windows CreateProcessAsUser function.*
- *The CreateProcessWithLogon function combines these into a single call,* which is how the Runas command launches processes under alternative tokens.

Access Token



PROF. HARJIT DHILLON

Impersonation

- Impersonation is a powerful feature Windows uses frequently in its security model.
- Windows also uses impersonation in its client/server programming model. For example, a server application can provide access to resources such as files, printers, or databases.
- Clients wanting to access a resource send a request to the server. When the server receives the request, it must ensure that the client has permission to perform the desired operations on the resource.
- For example, if a user on a remote machine tries to delete a file on an NTFS share, the server exporting the share must determine whether the user is allowed to delete the file.

Impersonation

- The obvious way to determine whether a user has permission is for the server to query the user's account and group SIDs and scan the security attributes on the file.
- This approach is tedious to program, prone to errors, and wouldn't permit new security features to be supported transparently.

Impersonation

- Thus, Windows provides impersonation services to simplify the server's job.
- Impersonation lets a server notify the SRM that the server is temporarily adopting the security profile of a client making a resource request.
- The server can then access resources on behalf of the client, and the SRM carries out the access validations, but it does so based on the impersonated client security context.
- A server has access to more resources than a client does and loses some of its security credentials during impersonation.
- However, the reverse can be true: the server can gain security credentials during impersonation.

Impersonation

- A server impersonates a client only within the thread that makes the impersonation request.
- Thread-control data structures contain an optional entry for an impersonation token.
- However, a thread's primary token, which represents the thread's real security credentials, is always accessible in the process' control structure.
- After the server thread finishes its task, it reverts to its primary security context.
- These forms of impersonation are convenient for carrying out specific actions at the request of a client and for ensuring that object accesses are audited correctly.

Security Descriptors and Access control

- Tokens, which identify a user's credentials, are only part of the object security equation.
- Another part of the equation is the security information associated with an object, which specifies who can perform what actions on the object.
- The data structure for this information is called a security descriptor.

Security Descriptors and Access control

- **A security descriptor consists of the following attributes:**
  - ***Revision number*** **The version of the SRM security model used to create the descriptor.**
  - ***Flags*** **Optional modifiers that define the behavior or characteristics of the descriptor.**
  - ***Owner SID*** **The owner's security ID**
  - ***Group SID*** **The security ID of the primary group for the object (used only by POSIX).**
  - ***Discretionary access control list (DACL)*** **Specifies who has what access to the object.**
  - ***System access control list (SACL)*** **Specifies which operations by which users should be logged in the security audit log and the explicit integrity level of an object.**

Account Rights and Privileges

- Many operations performed by processes as they execute cannot be authorized through object access protection because they do not involve interaction with a particular object.
- For example, the ability to bypass security checks when opening files for backup is an attribute of an account, not of a particular object.
- Windows uses both privileges and account rights to allow a system administrator to control what accounts can perform security-related operations.

Account Rights and Privileges

- A privilege is the right of an account to perform a particular system-related operation, such as shutting down the computer or changing the system time.
- An account right grants or denies the account to which it's assigned the ability to perform a particular type of logon, such as a local logon or interactive logon, to a computer.
- A system administrator assigns privileges to groups and accounts using tools such as the Active Directory Users and Groups MMC snap-in for domain accounts or the Local Security Policy Editor
- (%SystemRoot%\System32\secpol.msc).

Account Rights

- Account rights are not enforced by the security reference monitor, nor are they stored in tokens.
- The function responsible for logon is *LsaLogonUser. Winlogon, for example, calls the LogonUser API* when a user logs on interactively to a computer, and *LogonUser calls LsaLogonUser.*
- *LogonUser takes* a parameter that indicates the type of logon being performed, which includes interactive, network, batch, service, and Terminal Server client.

Account Rights

- In response to logon requests, the Local Security Authority (LSA) retrieves account rights assigned to a user from the LSA policy database at the time that a user attempts to log on to the system.
- LSA checks the logon type against the account rights assigned to the user account logging on and denies the logon if the account does not have the right that permits the logon type or it has the right that denies the logon type.

Account Rights

User Right	Role
Deny logon locally, Allow logon locally	Used for interactive logons that originate on the local machine
Deny logon over the network, Allow logon over the network	Used for logons that originate from a remote machine
Deny logon through Terminal Services, Allow logon through Terminal Services	Used for logons through a Terminal Server client
Deny logon as a service, Allow logon as a service	Used by the service control manager when starting a service in a particular user account
Deny logon as a batch job, Allow logon as a batch job	Used when performing a logon of type batch

Privileges

- The number of privileges defined by the operating system has grown over time.
- Unlike user rights, which are enforced in one place by the LSA, different privileges are defined by different components and enforced by those components.
- For example, the debug privilege, which allows a process to bypass security checks when opening a handle to another process with the *OpenProcess Windows API,* is checked for by the process manager.

Privileges

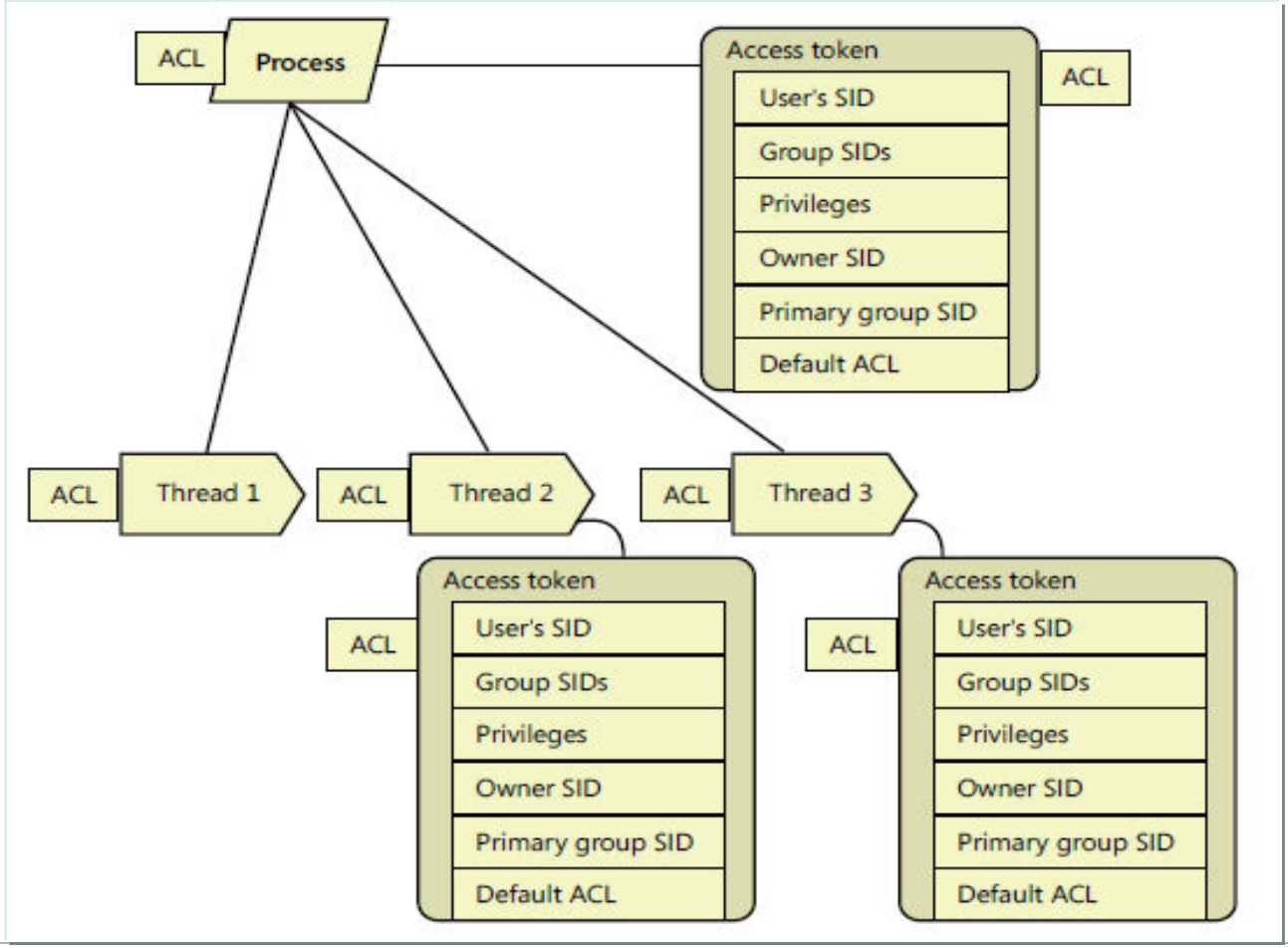
Privilege	User Right	Privilege Usage
SeAssignPrimaryTokenPrivilege	Replace a process-level token	Checked for by various components, such as *NtSetInformationJob*, that set a process' token.
SeAuditPrivilege	Generate security audits	Required to generate events for the Security event log with the *ReportEvent* API.
SeBackupPrivilege	Back up files and directories	Causes NTFS to grant the following access to any file or directory, regardless of the security descriptor that's present: READ_CONTROL, ACCESS_SYSTEM_SECURITY, FILE_GENERIC_READ, FILE_TRAVERSE Note that when opening a file for backup, the caller must specify the FILE_FLAG_BACKUP_SEMANTICS flag. Also allows corresponding access to registry keys when using *RegSaveKey*.
SeChangeNotifyPrivilege	Bypass traverse checking	Used by NTFS to avoid checking permissions on intermediate directories of a multilevel directory lookup. Also used by file systems when applications register for notification of changes to the file system structure.
SeCreateGlobalPrivilege	Create global objects	Required for a process to create section and symbolic link objects in the directories of the object manager namespace that are assigned to a different session than the caller.
SeCreatePagefilePrivilege	Create a pagefile	Checked for by *NtCreatePagingFile*, which is the function used to create a new paging file.
SeCreatePermanentPrivilege	Create permanent shared objects	Checked for by the object manager when creating a permanent object (one that doesn't get deallocated when there are no more references to it).
SeCreateSymbolicLinkPrivilege	Create symbolic links	Checked for by NTFS when creating symbolic links on the file system with the *CreateSymbolicLink* API.
SeCreateTokenPrivilege	Create a token object	*NtCreateToken*, the function that creates a token object, checks for this privilege.
SeDebugPrivilege	Debug programs	If the caller has this privilege enabled, the process manager allows access to any process or thread using *NtOpenProcess* or *NtOpenThread*, regardless of the process' or thread's security descriptor (except for protected processes).
SeEnableDelegationPrivilege	Enable computer and user accounts to be trusted for delegation	Used by Active Directory services to delegate authenticated credentials.

Super Privileges

- Several privileges are so powerful that a user to which they are assigned is effectively a “super user” who has full control over a computer.
- These privileges can be used in an infinite number of ways to gain unauthorized access to otherwise off-limit resources and to perform unauthorized operations.

Access Tokens of Processes and Threads

- Figure brings together the concepts covered by illustrating the basic process and thread security structures. In the figure, notice that the process object and the thread objects have ACLs, as do the access token objects themselves. thread 2 and thread 3 each have an impersonation token, whereas thread 1 uses the default process access token.

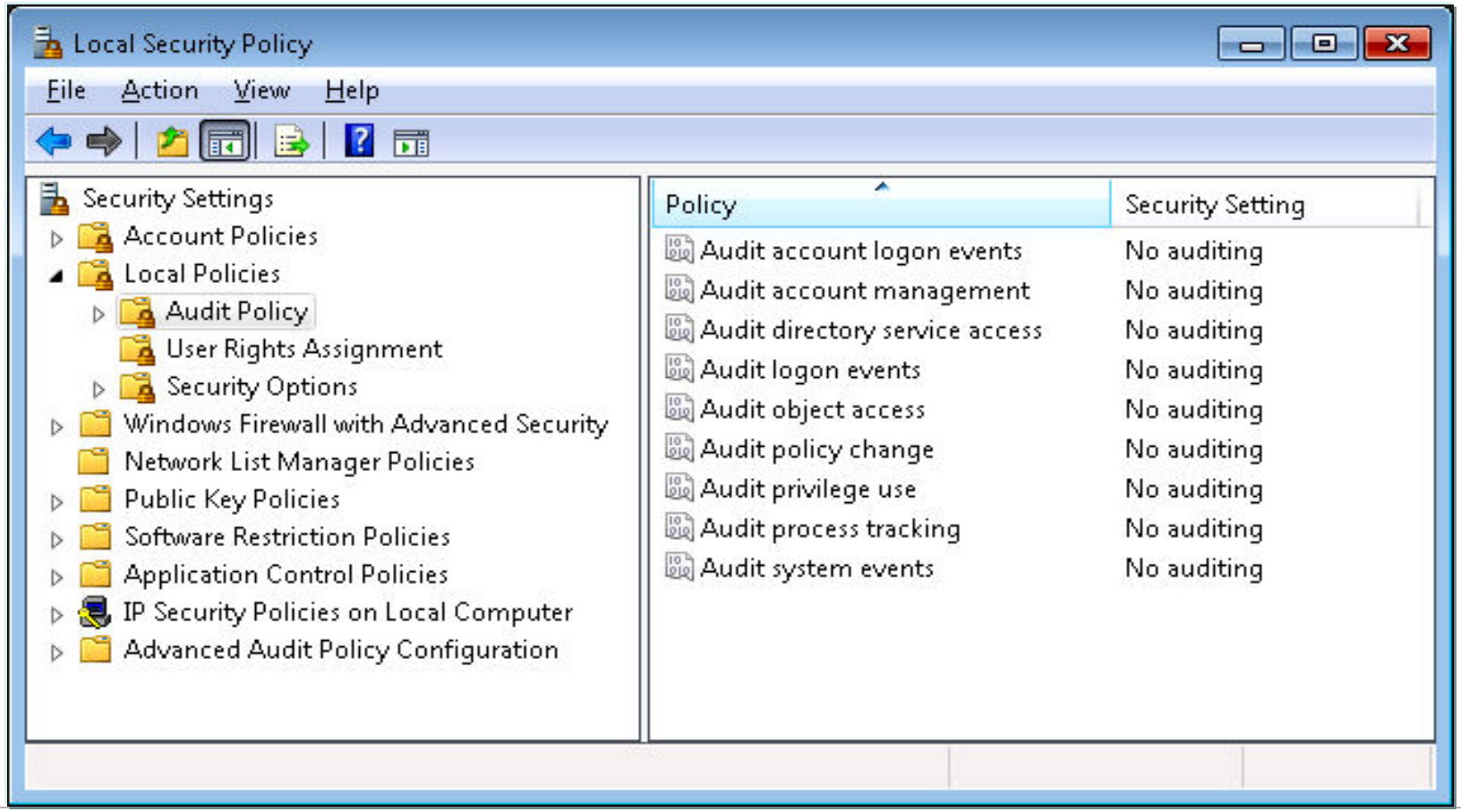


Security Auditing

- The object manager can generate audit events as a result of an access check, and Windows functions available to user applications can generate them directly. Kernel-mode code is always allowed to generate an audit event.
- Two privileges, SeSecurityPrivilege and SeAuditPrivilege, relate to auditing.
- A process must have the SeSecurityPrivilege privilege to manage the security Event Log and to view or set an object's SACL.
- Processes that call audit system services, however, must have the SeAuditPrivilege privilege to successfully generate an audit record.

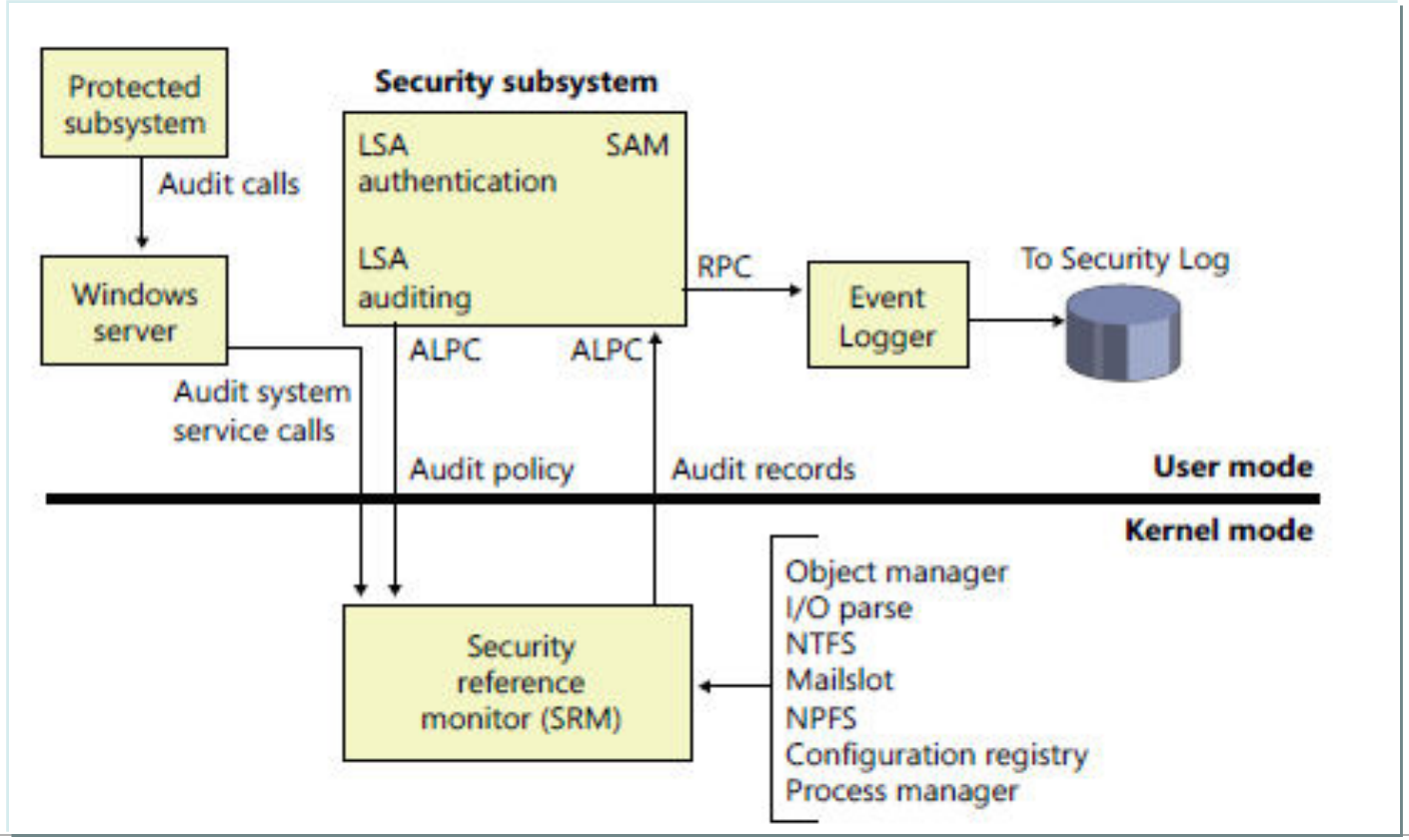
Security Auditing

- The audit policy of the local system controls the decision to audit a particular type of security event. The audit policy, also called the local security policy, is one part of the security policy LSASS maintains on the local system, and it is configured with the Local Security Policy Editor as shown in Figure



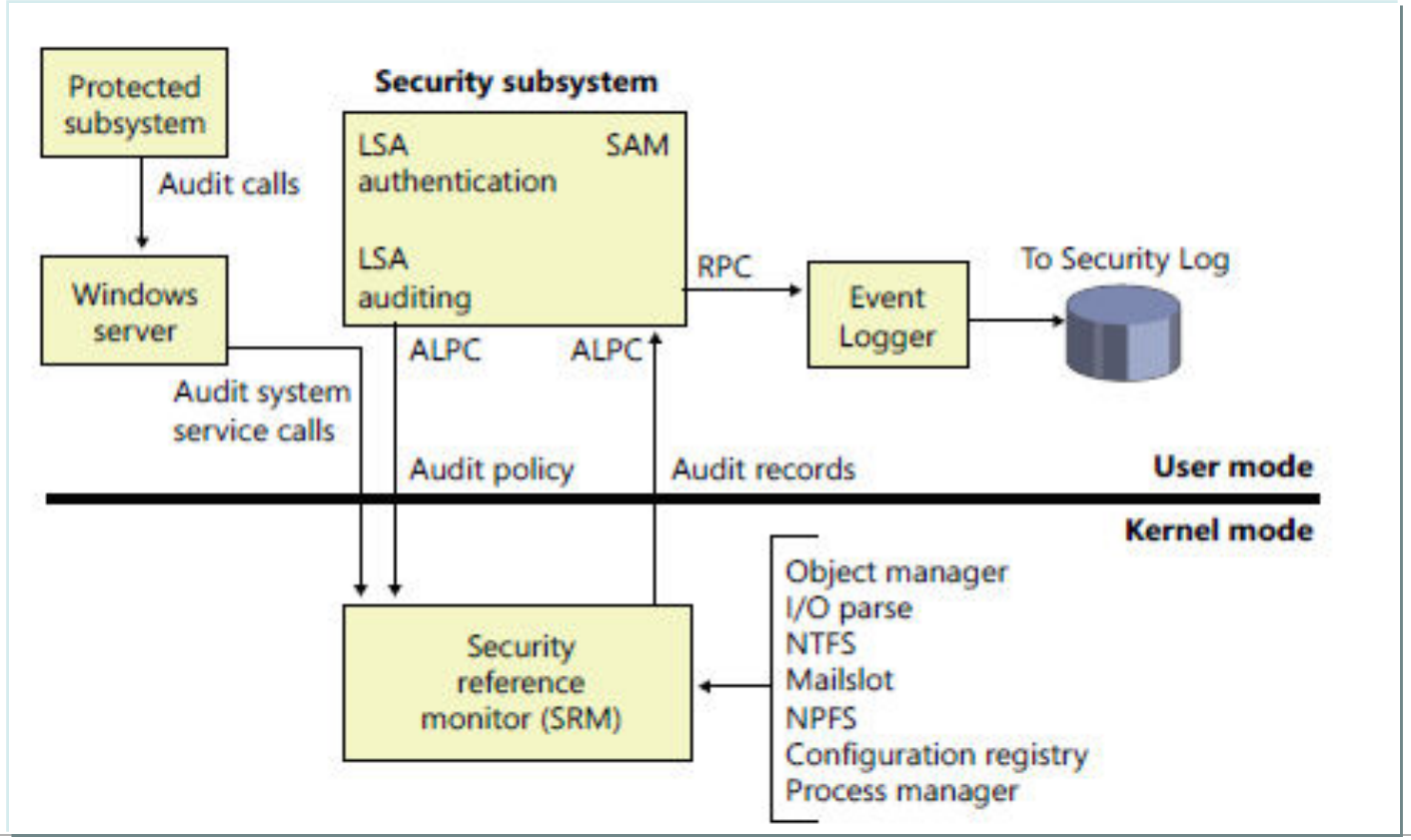
Flow of Security Audit Records

- LSASS sends messages to the SRM to inform it of the auditing policy at system initialization time and when the policy changes. LSASS is responsible for receiving audit records generated based on the audit events from the SRM, editing the records, and sending them to the Event Logger. LSASS (instead of the SRM) sends these records because it adds pertinent details, such as the information needed to more completely identify the process that is being audited.



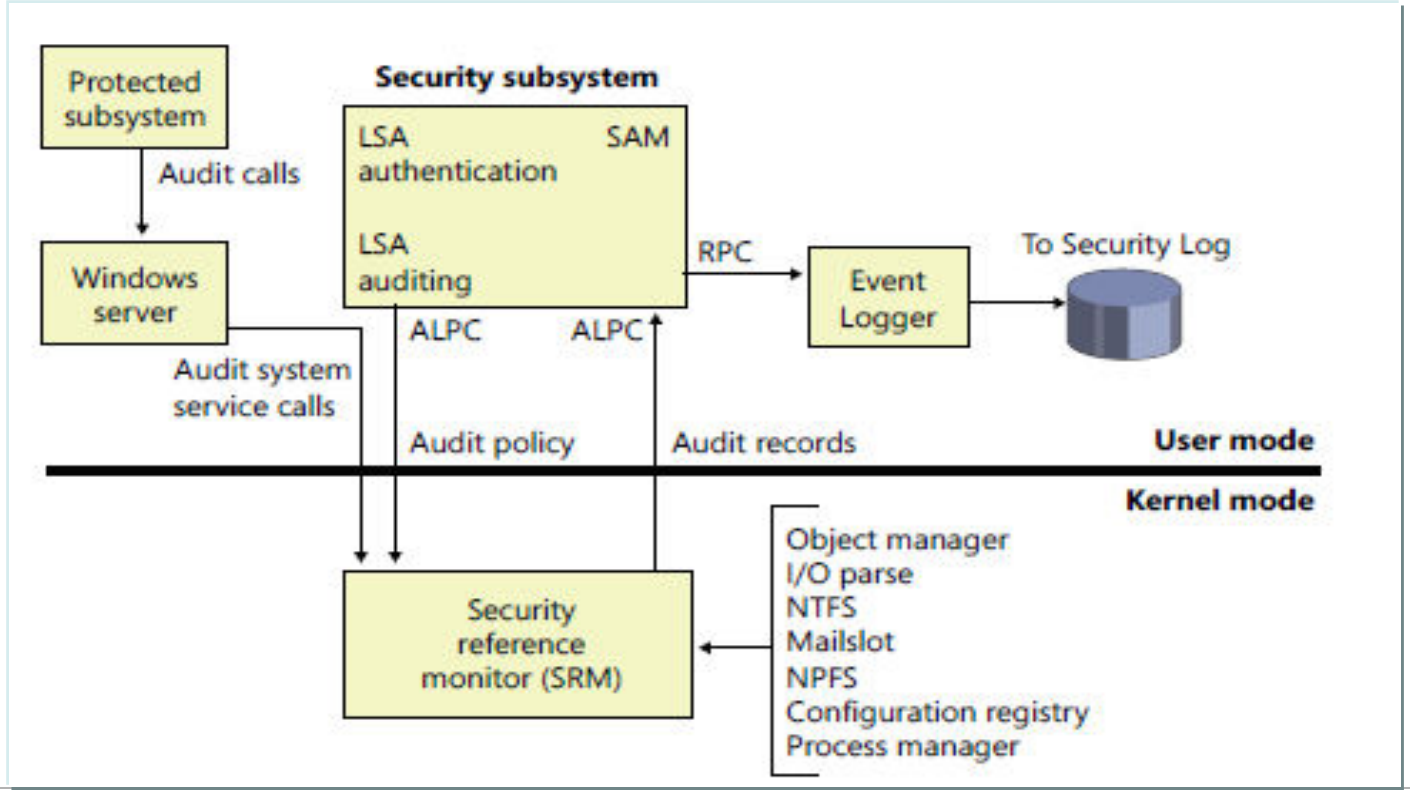
Flow of Security Audit Records

- The SRM sends audit records via its ALPC connection to LSASS. The Event Logger then writes the audit record to the security Event Log. In addition to audit records the SRM passes, both LSASS and the SAM generate audit records that LSASS sends directly to the Event Logger, and the AuthZ APIs allow for applications to generate application-defined audits. Figure depicts this overall flow.



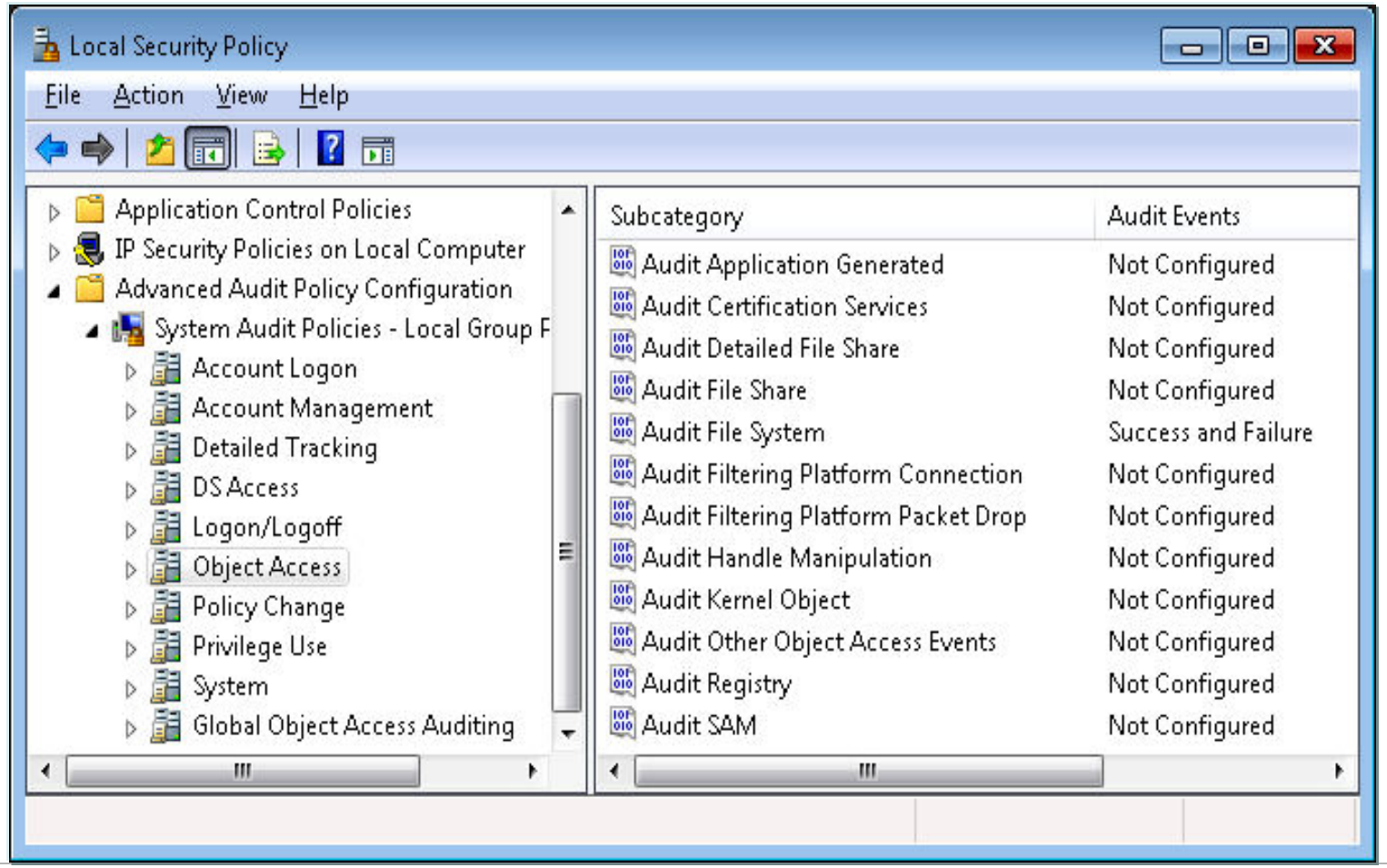
Flow of Security Audit Records

- Audit records are put on a queue to be sent to the LSA as they are received—they are not submitted in batches. The audit records are moved from the SRM to the security subsystem in one of two ways. If the audit record is small (less than the maximum ALPC message size), it is sent as an ALPC message. The audit records are copied from the address space of the SRM to the address space of the LSASS process. If the audit record is large, the SRM uses shared memory to make the message available to LSASS and simply passes a pointer in an ALPC message.



Advanced Audit Policy Settings

- In addition to the Audit Policy settings described previously, the Local Security Policy Editor offers a much more fine-grained set of audit controls under the Advanced Audit Policy Configuration heading, as shown in Figure



Logon

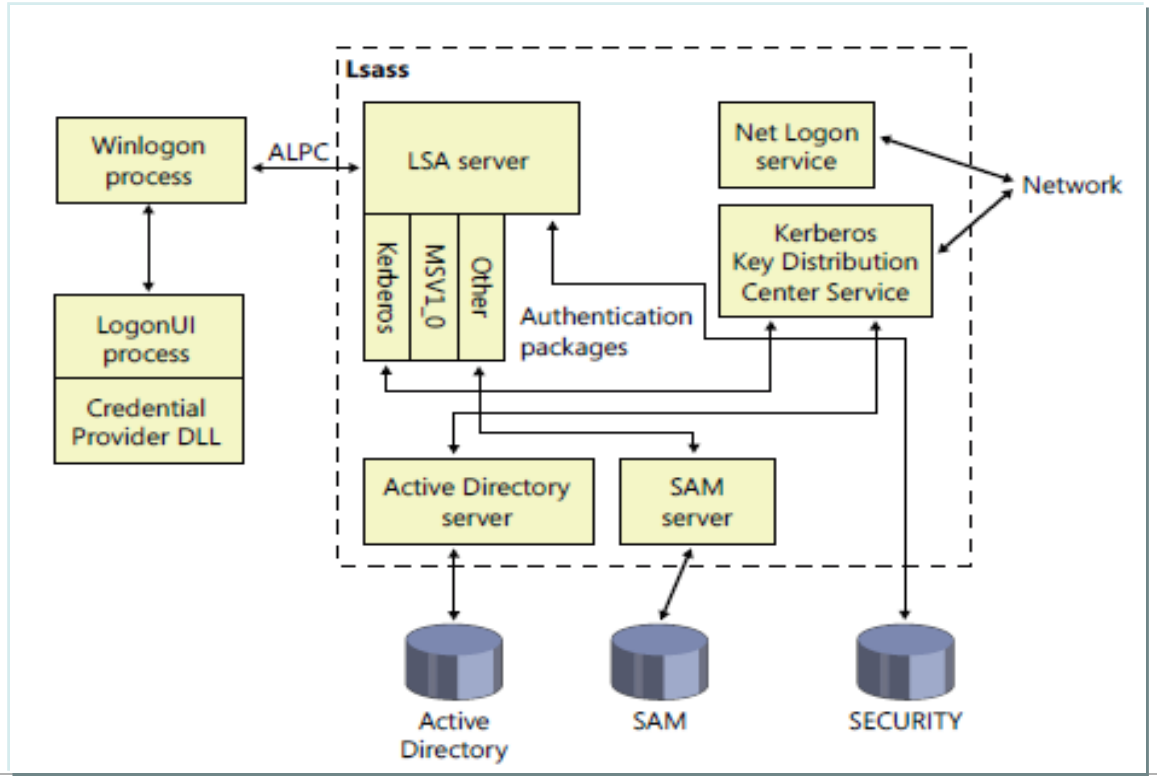
- Interactive logon (as opposed to network logon) occurs through the interaction of the logon process (Winlogon), the logon user interface process (LogonUI) and it credential providers, LSASS, one or more authentication packages, and the SAM or Active Directory.
- Authentication packages are DLLs that perform authentication checks.
- Kerberos is the Windows authentication package for interactive logon to a domain, and MSV1_0 is the Windows authentication package for interactive logon.

Logon

- Winlogon is a trusted process responsible for managing security-related user interactions.
- It coordinates logon, starts the user's first process at logon, handles logoff, and manages various other operations relevant to security, including launching LogonUI for entering passwords at logon, changing passwords, and locking and unlocking the workstation.
- The Winlogon process must ensure that operations relevant to security aren't visible to any other active processes.
- For example, Winlogon guarantees that an untrusted process can't get control of the desktop during one of these operations and thus gain access to the password.

Components involved in logon

- Winlogon is the only process that intercepts logon requests from the keyboard, which are sent through an RPC message from Win32k.sys. Winlogon immediately launches the LogonUI application to display the user interface for logon. After obtaining a user name and password from credential providers, Winlogon calls LSASS to authenticate the user attempting to log on. If the user is authenticated, the logon process activates a logon shell on behalf of that user. The interaction between the components involved in logon is illustrated in Figure

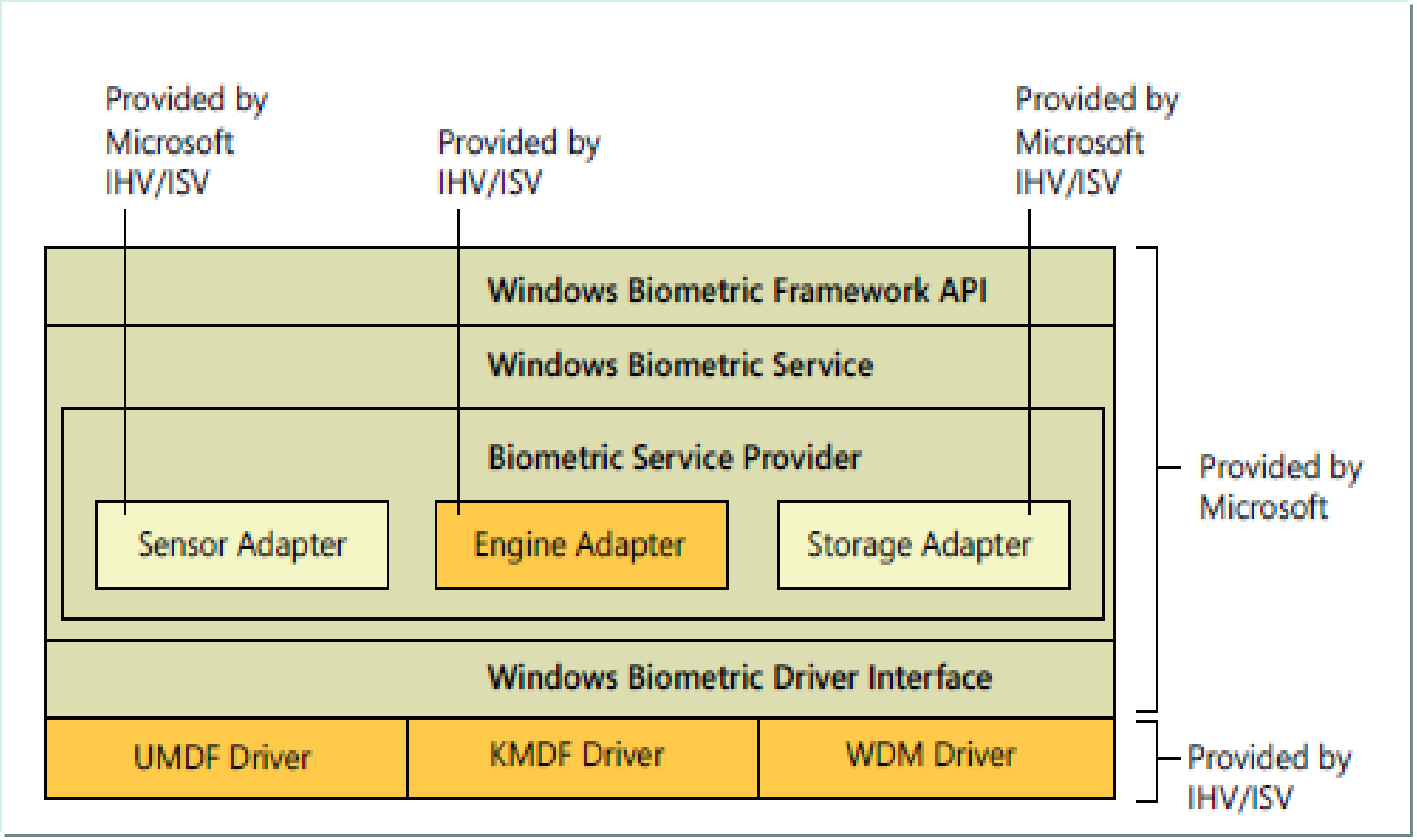


Biometric Framework for User Authentication

- Windows provides a standardized mechanism for supporting certain types of biometric devices—specifically, fingerprint scanners—to support user identification via a fingerprint swipe.
- Like many other such frameworks, the Windows Biometric Framework was developed to isolate the various functions
- involved in supporting such devices, so as to minimize the code required to implement a new device.

Windows Biometric Framework components and architecture

- The primary components of the Windows Biometric Framework are shown in Figure



Elevation

- Even if users run only programs that are compatible with standard user rights, some operations still require administrative rights.
- For example, the vast majority of software installations require administrative rights to create directories and registry keys in system-global locations or to install services or device drivers.
- Modifying system-global Windows and application settings also requires administrative rights, as does the parental controls feature.

Elevation

- It's important to be aware that UAC elevations are conveniences and not security boundaries.
- A security boundary requires that security policy dictate what can pass through the boundary.
- User accounts are an example of a security boundary in Windows, because one user can't access the data belonging to another user without having that user's permission.

Elevation

- Because elevations aren't security boundaries, there's no guarantee that malware running on a system with standard user rights can't compromise an elevated process to gain administrative rights.
- For example, elevation dialog boxes only identify the executable that will be elevated; they say nothing about what it will do when it executes.

Running with Administrator Rights

- Windows includes enhanced “run as” functionality so that standard users can conveniently launch processes with administrative rights.
- This functionality requires giving applications a way to identify operations for which the system can obtain administrative rights on behalf of the application, as necessary.
- To enable users acting as system administrators to run with standard user rights but not have to enter user names and passwords every time they want to access administrative rights, Windows makes use of a mechanism called Admin Approval Mode (AAM).

Application Identification (AppID)

- Historically, security decisions in Windows have been based upon a user's identity (in the form of the user's SID and group membership), but a growing number of security components (AppLocker, firewall, antivirus, antimalware, Rights Management Services, and others) need to make security decisions based upon what code is to be run.
- In the past, each of these security components used their own proprietary method for identifying applications, which led to inconsistent and overly-complicated policy authoring.
- The purpose of AppID is to bring consistency to how the security components recognize applications by providing a single set of APIs and data structures.

AppLocker

- ***New to Windows 7 and Windows Server 2008/R2 (Enterprise and Ultimate editions)*** **is a feature known as AppLocker, which allows an administrator to lockdown a system to prevent unauthorized programs from being run.**
- **Windows XP introduced Software Restriction Policies (SRP), which was the first step toward this capability, but SRP suffered from being difficult to manage, and it couldn't be applied to specific users or groups. (All users were affected by SRP rules.)**

AppLocker

- AppLocker is a replacement for SRP, and yet coexists alongside SRP, with AppLocker's rules being stored separately from SRP's rules.
- If both AppLocker and SRP rules are in the same Group Policy object (GPO), only the AppLocker rules will be applied.

AppLocker

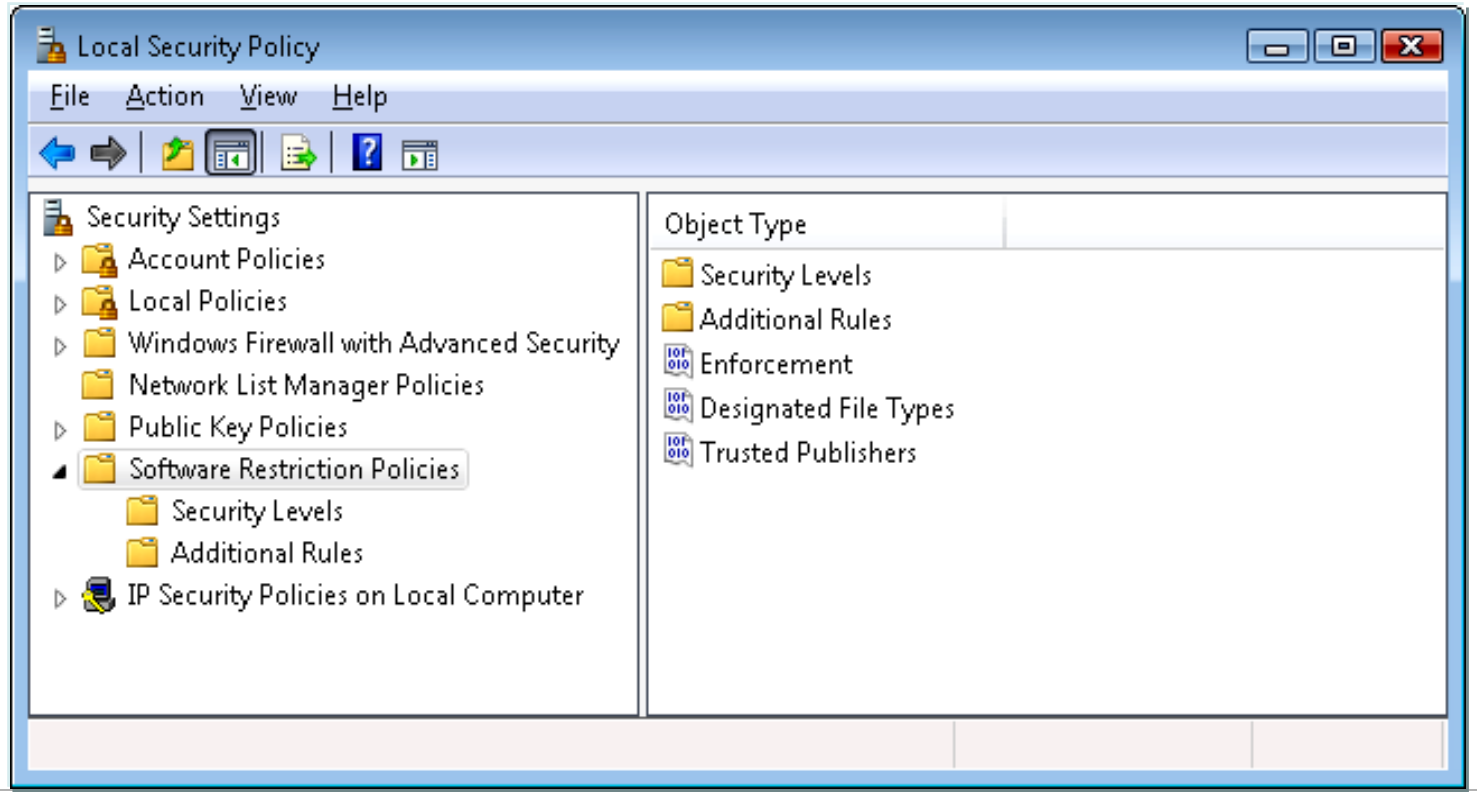
- Another feature that makes AppLocker superior to SRP is AppLocker's auditing mode, which allows an administrator to create an AppLocker policy and examine the results (stored in the system event log) to determine whether the policy will perform as expected—without actually performing the restrictions.
- AppLocker auditing mode can be used to monitor which applications arebeing used by one, or more, users on a system.

AppLocker

- AppLocker allows an administrator to restrict the following types of files from being run:
  - Executable images (.EXE and .COM)
  - Dynamic-Link Libraries (.DLL and .OCX)
  - Microsoft Software Installer (.MSI and .MSP) for both install and uninstall
  - Scripts
  - Windows PowerShell (.PS1)
  - Batch (.BAT and .CMD)
  - VisualBasic Script (.VBS)
  - Java Script (.JS)

Software Restriction Policies

- **Windows also contains a user-mode mechanism called Software Restriction Policies that enables administrators to control what images and scripts execute on their systems. The Software Restriction Policies node of the Local Security Policy Editor, shown in Figure serves as the management interface for a machine's code execution policies, although per-user policies are also possible using domain group policies.**



Software Restriction Policies

- **Several global policy settings appear beneath the Software Restriction Policies node:**
  - ***The Enforcement policy*** **configures whether restriction policies apply to libraries, such as DLLs, and whether policies apply to users only or to administrators as well.**
  - ***The Designated File Types policy*** **records the extensions for files that are considered executable code.**
  - ***Trusted Publishers*** **control who can select which certificate publishers are trusted.**

SYSTEM HARDENING BEST PRACTICES

- Disable unnecessary services
- Develop a process for updating *all* software
- Change default port numbers
- Use network and host-based firewalls
- Require IPSec
- Place Internet servers in perimeter networks
- Use physical security
- Restrict removable media
- Backup application-specific information

SYSTEM HARDENING BEST PRACTICES (CONT.)

- Audit backups and restores
- Rename default user accounts
- Develop security requirements for application-specific user databases
- Monitor each server role for failures
- Read security guides

HARDENING DOMAIN CONTROLLERS

- A compromised domain controller can lead to compromises of domain members
- Domain controllers can be identified with a DNS query
- Avoid storing application data in Active Directory
- Create a separate security group for users with privileges to backup domain controllers
- Use source-IP filtering to block domain requests from external networks

REQUIRE DOMAIN CONTROLLER SERVICES

- File Replication Service
- Intersite Messaging
- Kerberos Key Distribution Center
- Netlogon
- Remote Procedure Call (RPC) Locator
- Windows Management Instrumentation
- Windows Time

BEST PRACTICE

- DISK PARTITIONS ARE FORMATTED WITH NTFS
- CHANGE SYSTEM LOGGING SETTINGS
- DISABLE INDEXING SERVICE
- PROTECT FILE SHARES
- DISABLE USER SWITCHING
- USE SOFTWARE RESTRICTION POLICIES
- DISABLE UNNECESSARY SERVICES
- KEEP UP-TO-DATA on the Latest Security Updates

Encryption Methods Microsoft Windows Supports

- LOCAL SYSTEM
  - BitLocker for encrypting entire volumes
  - Encrypting File System (EFS) for encrypting files and folders

- TRANSIT TRAFFIC
  - • Secure networking protocols
  - • Digital certificates
  - • Public Key Infrastructure
  - • Virtual Private Networks

What is Encrypting File System (EFS)

- The Microsoft Windows Encrypting File System (EFS) is feature built into the file system
- It lets you encrypt designated files on a local computer so that no other user can access your data.
- When a file is encrypted, EFS automatically decrypts the file for use and re-encrypts the file when it is saved.
- EFS is particularly useful for protecting data on a computer that might be physically stolen, such as a laptop.

What It Doesn't Protect or Prevent

It does NOT provide encryption to files that are:

- Sent via email
- Kept on a separate flash drive/thumb drive/USB drive/floppy disk
- Moved over the network via shared folders (CIFS/AFS)
- System and page file
- Compress Files
- Files moved into folder set to encrypt all files
- Files form being deleted

- When you are about to move an encrypted file, Windows will warn you that you will lose your EFS encryption.
- *Keep in mind that whenever you move a file off of your computer, it is probably no longer protected by EFS.*

BitLocker™ Drive Encryption

- BitLocker™ Drive Encryption gives you improved data protection on your Windows Vista and Windows Server codenamed “Longhorn” systems
  - Notebooks – Often stolen, easily lost in transit
  - Desktops – Often stolen, difficult to safely decommission
  - Servers – High value targets, often kept in insecure locations
  - All three can contain very sensitive IP and customer data
- Designed to provide a transparent user experience that requires little to no interaction on a protected system
- Prevents thieves from using another OS or software hacking tool to break OS file and system protections
  - Prevents offline viewing of user data and OS files
  - Provides enhanced data protection and boot validation through use of a Trusted Platform Module (TPM)

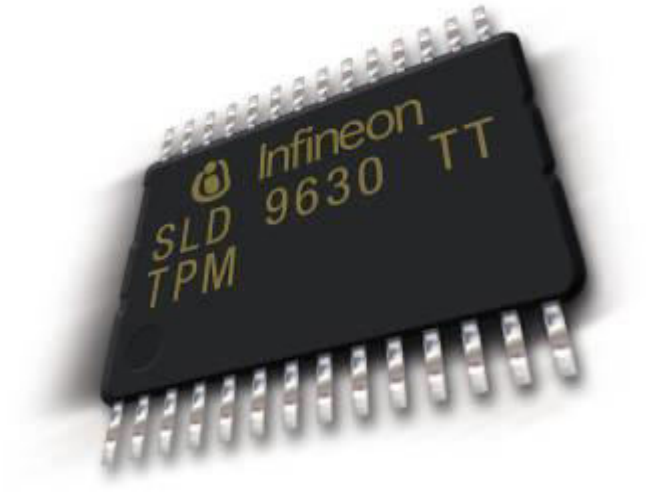
BitLocker™ And TPM Features

- **BitLocker™ Drive Encryption**
  - Encrypts entire volume
  - Uses Trusted Platform Module (TPM) v1.2 to validate pre-OS components
  - Customizable protection and authentication methods
- **Pre-OS Protection**
  - USB startup key, PIN, and TPM-backed authentication
- **Single Microsoft TPM Driver**
  - Improved stability and security
- **TPM Base Services (TBS)**
  - Enables third party applications
- **Active Directory Backup**
  - Automated key backup to AD server
  - Group Policy support
- **Scriptable Interfaces**
  - TPM management
  - BitLocker™ management
  - Command-line tool

What Is A Trusted Platform Module (TPM)?

Smartcard-like module on the motherboard

- **Protects secrets**
- **Performs cryptographic functions**
  - **RSA, SHA-1, RNG**
  - **Meets encryption export requirements**
- **Can create, store and manage keys**
  - **Provides a unique Endorsement Key (EK)**
  - **Provides a unique Storage Root Key (SRK)**
- **Performs digital signature operations**
- **Holds Platform Measurements (hashes)**
- **Anchors chain of trust for keys and credentials**
- **Protects itself against attacks**



BitLocker vs EFS

BITLOCKER	ENCRYPTING FILE SYSTEM (EFS)
Encrypts all files on the selected volume	Encrypts only selected files and folders
Either on or off for all users	Encrypts files based on user actions—each user can encrypt files or folders individually
Uses TPM or USB key as part of the authentication process	Does not require any special hardware
Must be administrator to turn BitLocker on or off	Any user can choose to encrypt files or folders

Windows 10 Security Enhancements

Windows 7	Windows 10	
Malware starts before Windows, takes control, and evades detection	Helps prevent malware from compromising system before OS and defenses can start	Windows Trusted Boot
Passwords are easily stolen Multi Factor authentication too hard	Passwords can be replaced with biometrics and easy to use multi-factor authentication	Windows Hello
User credentials are easily stolen on companies networks	User credentials are protected using hardware based virtualization/isolation	Credential Guard
Malware can bypass anti-virus and app control solutions	Next Gen app control and OS hardening gives IT better control of what runs in their environment	Device Guard
Users and apps can leak business data without restriction	Data separation and containment capabilities help prevent accidental data leaks	Enterprise Data Protection
3rd party solutions required to detect targeted attacks on devices	Helps detect and respond to breaches with built in behavioral sensors and cloud based analytics	Windows Defender ATP

Windows – New Security Features