

Chapter

1

Windows Security

It is important that you understand Windows security to gain insight into how Windows Installer works. If you want to understand elevated privileges, you must understand the Windows security basics. The same applies for understanding the various types of custom actions and when you should use one type instead of another. Windows security also comes into play when you are installing software to a locked-down system. Many of the subsequent chapters in this book require an understanding of material provided in this chapter.

Because it is an expansive subject, a complete discussion of Windows security is outside the scope of this book. This chapter provides only an introduction to the subject and covers those areas important to software installation.

Security Environment Overview

The authentication and authorization processes are the basis for the security approach implemented by the Windows 2000 and Windows XP operating systems. Each user—sometimes referred to as a principal—presents a set of credentials to the system and the authentication process ensures that the user is who they claim to be. After a user's identity has been authenticated, the user is authorized certain privileges and rights on the system. The privileges granted to a user pertain to the performance of a subset of system-wide actions that are possible. Additionally, the user is given rights to access securable resources on the local machine and/or on the network.

A user can be a person, a Windows service, or another computer, and each of these entity types can simultaneously access resources on a particular computer or a network. Each entity that accesses the resources on a computer does so from within a logon session unique to that particular entity. Any process launched by an entity runs under the context for that entity.

User and Group Accounts

Before any entity can log on to a computer or network, a user account needs to have been created—with the exception of the LocalSystem account. The operating system and most Windows services run under this account on the local computer. Due to the importance of the LocalSystem account to the installation of software by Windows Installer, this special account is discussed in greater detail later in this chapter.

You must have local administrator privileges to create a user account on a local machine. To create a user account on a network domain, you must also have domain administrator privileges. When you install Windows 2000 Professional or Windows XP Professional, a number of built-in groups and users are created. These built-in accounts, along with their descriptions, are shown in Table 1-1. Table 1-2, later in the chapter, shows the default privileges of each of these built-in group and user accounts.

Table 1-1: *Built-In Group and User Accounts.*

Account Name	Type	Description
Administrators	Group	Members of the Administrators group have complete and unrestricted access to the computer/domain.
Power Users	Group	The members of this group have many of the privileges that an administrator has. Power Users have read/write permission to other parts of the system, in addition to their own profile folders. Power Users can install applications and perform many administrative tasks. A Power User can run legacy applications that have not been certified by Windows 2000.
Backup Operators	Group	The members of this group can back up and restore files on a computer. They are able to override security restrictions for the sole purpose of backing up or restoring files, regardless of the permissions that protect those files. A Backup Operator can also log on to a computer and shut it down, but cannot change security settings.
Users	Group	The members of this group have read-only permission for most parts of the system and read/write permission in their own profile folders. A user is unable to read other users' data, install applications that require modification of system directories, or perform administrative tasks.

(Continued)

Table 1-1: *(Continued)*

Account Name	Type	Description
Guests	Group	Members of this group are allowed occasional or one-time access to a workstation's built-in Guest account. Members of this group have, by default, the same access as members of the Users group.
Administrator	User	This user account is automatically a member of the Administrators group and is the first account created when installing Windows 2000 or Windows XP.
Guest	User	This user account is an automatic member of the Guests group. This account is disabled by default.

A number of additional built-in group and user accounts are created when you install a server operating system.

When you create a user account, you must define the basic credentials, which users present to the system every time they log on. In general, these credentials consist of a human-readable account name string and a password. Optionally, you can add the user's full name and a description string to the user account information. When the system creates the user account, the above information is entered either into the Security Accounts Manager (SAM) database or into Active Directory in the case of a Windows 2000 network domain account. When the account is created, the system defines a machine-readable name for the account called the security identifier (SID). This SID is also entered into the SAM database or Active Directory.

Another type of account, called a group account, is a mechanism provided to simplify the management of the security settings on a computer or network. As an administrator, you add user accounts to a group account and you allow the user accounts to have all the privileges assigned to the group. Consequently, you do not need to configure the privileges for each individual user account. When you create a

group account, the system also provides a machine-readable name for this group (also a SID). Group accounts are a management mechanism only; they do not have passwords and you cannot log on to a system using a group account name.

When you create a new user account, it is automatically added to the built-in Users group defined by the operating system. As an administrator, you can add a user account to any of the other built-in groups, or you can create a custom group and give this new group the privileges as required. You can also remove user accounts from any and all groups. A more detailed discussion about users, groups, and privileges is provided later in this chapter.

Passwords

The user account name and the password used to create the account represent the credentials that the user must present to the system at log on. The creation of secure passwords is critical to preventing non-authorized users from gaining access to either the local system or the network. An administrator can define certain policies forcing users to create secure passwords.

As an administrator, you can specify the minimum length of a password, the minimum age of a password, or the complexity of the password created. It is normally recommended that passwords consist of a combination of lowercase and uppercase letters, numbers, and symbols. The longer a password is, the more secure it is considered. Group Policy settings are used to enforce the password requirements on both the local machine and on the network. However, domain controller Group Policy settings override local settings.

Every user has two passwords if their computer is hooked into a network domain. One password is encrypted and stored in the SAM database on the local machine. The second password is encrypted and stored in the Active Directory on the domain controller. The single sign-on strategy allows users access to resources on the local machine in addition to those on the network without having to continually prove their identity by repeatedly entering their account name and password.

Single sign-on forces users to present their credentials once for authentication. The system then provides transparent authentication for all the resources the user subsequently tries to access. When you log on to your workstation, you normally have the choice to log on to a domain or on to your local machine. When you log on to a

domain, you still expect to have access to your local machine. This is made possible by including the domain user account as a member of a group on the local machine. In this way you can have, for example, administrative privileges on your local machine and only user privileges on the network.

Security Identifiers (SIDs)

When you create a user or group account on a local computer or a domain controller, a unique machine-readable security identifier (SID) is created for that account. The operating system refers to this identifier when it determines whether the logged-on user has the necessary privileges to perform a certain system-wide operation or whether the user has the permissions to access a particular securable object.

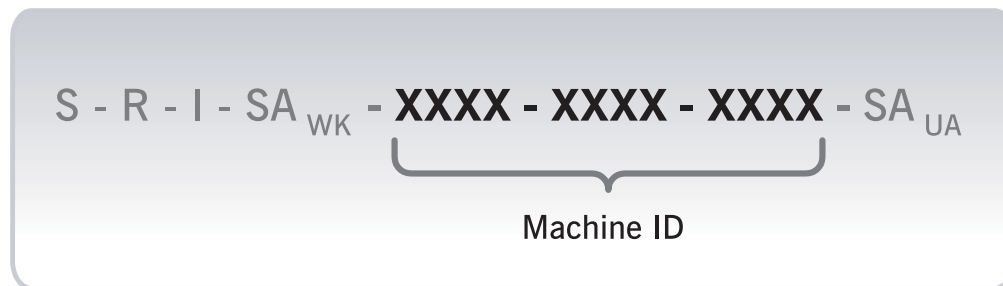


Figure 1-1: *The SID format.*

This SID is a variable-length binary data structure consisting of several elements. When you see a SID written in the registry, you are seeing this SID in a particular standardized string notation. The general format of the string notation for a SID is shown in Figure 1-1. This is a limited version of the SID as you are most likely to encounter it in the registry. A SID in its most general form can be of unlimited length, but this is not how you will see them in the normal process of installing software.

Each element of a SID as shown in Figure 1-1 is described in the following list:

- S:** Indicates that what follows is a security identifier.
- R:** Represents the revision level of the SID binary format. The revision is currently 1 and has been this value since Windows NT 3.1.

I: A 48-bit value defining the agent or authority that issued the SID. For all users and groups that are created on the local system or on the domain, the authority is called the NT Security Authority. The value that identifies this authority is 0x000000000005. There are five other authority values that have been defined, but you will rarely encounter these. You can find more information about these other values in the MSDN Library.

SA_{WK}: The remaining elements of a SID are 32-bit values called sub-authority identifiers. The first sub-authority value can be considered a well-known sub-authority and the value that you are most interested in is 0x00000015. The decimal equivalent of this is 21, and it is used as the starting point for the creation of all group and user security identifiers on the local machine or on a domain. You can find more information about the other values used for this first sub-authority element in the MSDN Library.

Machine ID: When you install one of the Windows NT–based operating systems, the setup program assigns a 96-bit unique identifier to the computer. For each group or user account that is created, this unique machine identifier is used to populate the sub-authorities in the second through fourth sub-authority positions of the SID. This is shown in Figure 1-1.

SA_{UA}: For group and user accounts, there is a locally unique value that is appended as the last sub-authority element of the SID. This value starts at 1000 and is incremented by 1 for each new group or user created. These values are unique only on the machine where the group or user is created and, because of this, it is normally called a relative identifier or RID. There are a number of predefined accounts that are created when you install the operating system. Among these are the local administrator’s account—which always uses a RID of 500—and the guest account that always uses a RID of 501.

What is described above is the structure of a SID created for group and user accounts that you create on a local machine or on a domain controller. Not all SIDs require the incorporation of the machine ID and the final RID. The operating system defines a number of well-known SIDs that are not tied to the unique machine ID and RID. You can find a discussion of these well-known SIDs in the MSDN Library. There is one particular well-known SID of interest here: the SID that defines the LocalSystem account. The SID for this account is the same on all machines and is as follows:

S-1-5-18

The LocalSystem account does not need an account name or a password. This type of account is discussed in more detail later in this chapter. Most Windows services run under the LocalSystem account.

Another well-known SID that you will often encounter is the SID that represents the Everyone group. This SID is used when specifying that all users can access a securable object. This SID is not used to define an account on a system. The value of this SID is as follows:

S-1-1-0

Window Stations and Desktops

A window station is a secure kernel object. This kernel object contains a clipboard, a set of global atoms, and one or more desktop objects. There is a special window station, named WinSta0, assigned to the logon session of the interactive user. This special window station also contains the keyboard, mouse, and display device. The interactive window station is visible to and can receive input from the user. All other window stations are non-interactive, meaning the window station cannot be made visible to the user and therefore cannot receive user input.

The main purpose of a window station is to provide a secure boundary around a set of processes. In the case of the interactive window station, this secure boundary eliminates the need for each application window to check the security of every other window displayed by a process running in the same window station. This approach enhances system performance, and also simplifies the programming necessary to create an application that runs in the interactive window station.

One object type contained in a window station is the desktop object. This is a secure type of object that has a display surface and is a container for windows, menus, and other items. A window station can have many desktop objects, but this makes sense only for the interactive window station. In the interactive window station, three desktops are defined automatically.

The first desktop that you see in the interactive window station is the one displayed when you press Ctrl+Alt+Del to enter your logon credentials. This same desktop is displayed when you press Ctrl+Alt+Del to launch the Task Manager after you have

logged on. You see another desktop with the shell after the log on is successful. Finally, a different desktop is displayed when a screensaver is running.

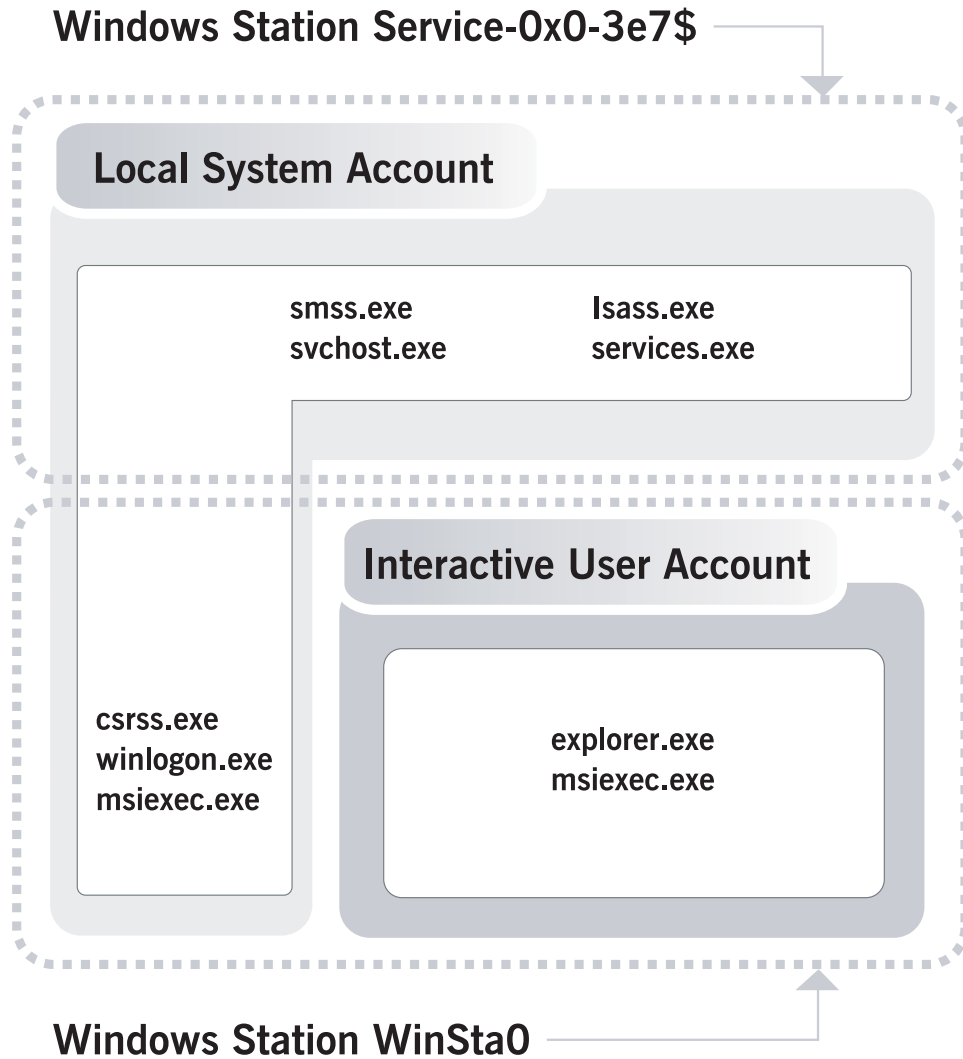


Figure 1-2: *The two standard window stations.*

Normally, a Windows service runs in a non-interactive window station under the LocalSystem account, and thus it is not possible for a service to interact with the user directly. However, when a Windows service is identified as needing to interact with

the user, this type of service is marked as interactive in the registry. The Service Control Manager (SCM) still runs this type of service in the LocalSystem security context, but attaches it to the interactive window station. This situation is shown in Figure 1-2 for the *msiexec.exe* service that is running under the LocalSystem account, but is attached to the interactive window station WinSta0. The reason that the *msiexec.exe* service is attached to the interactive window station is so it can display error messages to the user when something goes wrong during an installation. Because the LocalSystem account has access to all objects on the system, this potentially provides a user with access to all securable objects. This concept is important to the understanding of how the Windows Installer service can permit users to install software without administrative privileges. Figure 1-2 will be revisited later in this chapter when the mechanism used to give Windows Installer packages elevated privileges is discussed.

There are a few interesting things shown in Figure 1-2. First, the name of the non-interactive window station associated with the LocalSystem account is a fixed name always given to this object. The number 0x03e7 (999 decimal) that makes up part of the name of non-interactive window station is the hard-coded identifier for the LocalSystem account logon session. Also, notice that the executable *msiexec.exe* is shown as running in both the LocalSystem account and running under the interactive user account. The Windows Installer engine, *msiexec.exe*, runs in the LocalSystem account as a Windows service. When *msiexec.exe* runs under the account of the interactive user, it runs as a client to the Windows service.

The Interactive Logon Session

There are several logon session types—the most familiar of which is the interactive logon session. Because this logon session type is something that you are familiar with, this is an excellent place to start understanding Windows security concepts. The process of logging on interactively to a Windows 2000 or Windows XP operating system is shown in Figure 1-3. Working through this process will introduce you to a number of definitions important to your understanding of Windows security.

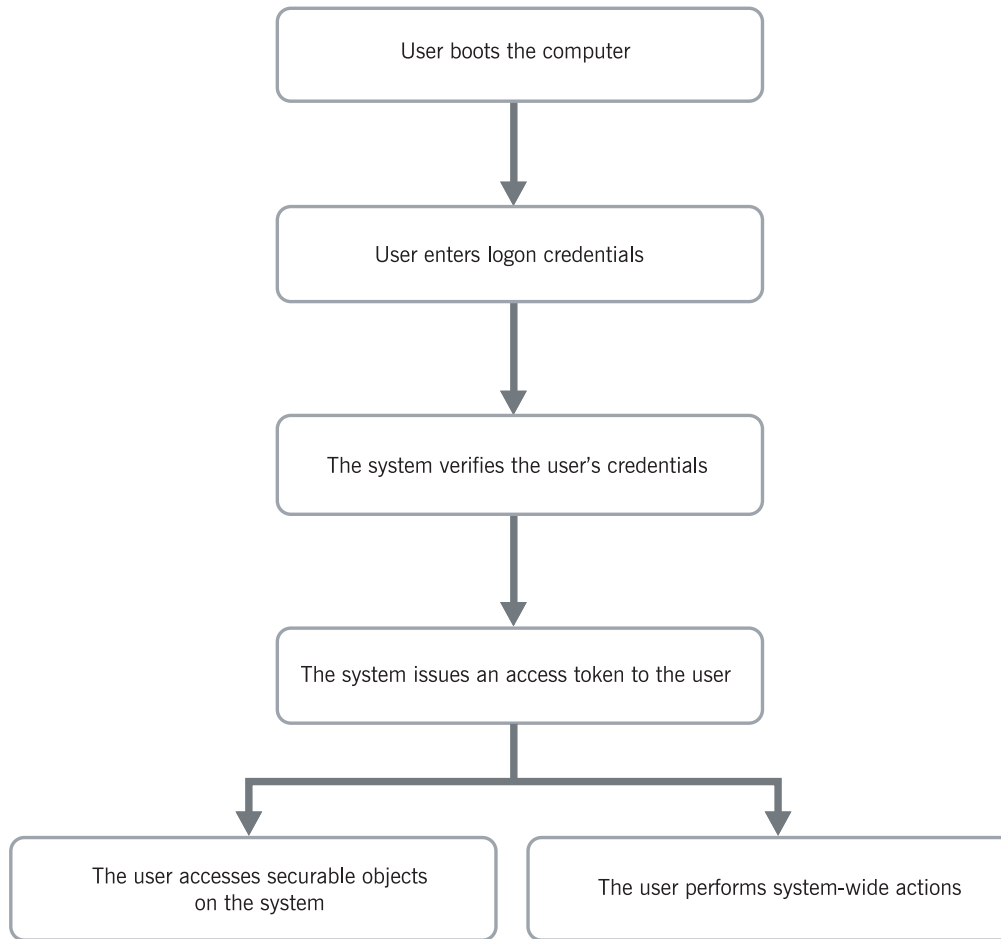


Figure 1-3: *An interactive logon session.*

The following sections discuss each item shown in Figure 1-3.

Starting the Operating System

This discussion focuses on how the system gets to the point where you can log on. The actions that take place to get to this point are shown below as steps. The details of what goes on during boot up of the operating system are not covered.

Step 1: The computer's BIOS loads the code found in the master boot record and then transfers control to this code.

Step 2: The master boot record code scans the primary partition table until it locates a bootable partition. From the first sector of this bootable partition, it loads into memory the code contained there and transfers control to this code.

Step 3: The boot sector code examines the root directory of the bootable partition, finds the Ntldr system file, and loads it into memory.

Step 4: The Ntldr system file switches from real mode to protected mode and then enables paging. The Ntldr system file then reads the *Boot.ini* file and presents a menu if one is defined. When the operating system to be booted is determined, Ntldr loads the *Ntoskernel.exe*, *Hal.dll*, and device drivers required for the boot. Ntldr then calls the main function of *Ntoskernel.exe* and ends its part of the boot process.

Step 5: The *Ntoskernel.exe* file initializes the executive subsystem. This executable also initializes the boot-start device drivers and the system-start device drivers. Finally, *Ntoskernel.exe* launches *Smsc.exe*, which is the Session Manager.

Step 6: The Session Manager performs a number of initialization tasks including loading the known DLLs, initializing the registry, and creating the system environment variables. The Session Manager then starts the Win32 subsystem, which includes running *Csrss.exe*. Finally, the Session Manager starts the Winlogon process by running *Winlogon.exe*.

Step 7: The Winlogon process loads the Graphical Identification and Authentication (GINA) DLL. The default GINA DLL provided by Microsoft is named *Msgina.dll*. The Winlogon process also loads the Local Security Authentication Subsystem (Lsass) by launching *Lsass.exe* and the Service Control Manager (SCM) by launching *Services.exe*. *Winlogon.exe* creates and opens the interactive window station and then creates and opens the three standard desktops—the Winlogon desktop, the default desktop, and the screensaver desktop. A connection is now made with the Lsass subsystem to enable authentication of a user's credentials. Finally, a special window is registered against the keyboard input that will be associated with the dialog that asks for the user's credentials. The *Winlogon.exe* and the *Csrss.exe* processes are attached to the

interactive window station, although they are running in the LocalSystem account logon session.

Step 8: The Service Control Manager loads and initializes all the auto-start device drivers and Win32 services.

After these steps, the system displays the Welcome to Windows dialog box and waits for the user to press the Ctrl+Alt+Delete keyboard shortcut. The display of this dialog box coincides closely with the completion of Step 8. The user logon process is discussed in the next section.

Creating the Interactive Logon Session

At this point, the system has been booted and is waiting for the user to start the logon process. All processes that run in the LocalSystem account have been started if they are of the auto-start type. Remember that the LocalSystem account is special in that it requires neither an account name nor a password. The processes that are running under the LocalSystem account at this point are attached to a special non-interactive window station except for those mentioned above (see Figure 1-2).

The logon process begins when the user presses the Ctrl+Alt+Delete keyboard shortcut. This key combination is referred to as the Secure Attention Sequence (SAS). The *winlogon.exe* process is the only one that can capture this sequence. This functionality prevents another process from masquerading as the Winlogon process and capturing a user's password. When the SAS is entered, *winlogon.exe* calls into the GINA DLL for the dialog box where the user will enter their account name and password. The GINA DLL is the source of all user interface dialog boxes that are displayed from the Winlogon process. As soon as the user's credentials are entered and the OK button is clicked, *winlogon.exe* sends these credentials to another process to be authenticated. The password entered is encrypted before it is passed on for authentication.

User Authentication

Winlogon passes the credentials to one or more authentication packages. A handle to each of the registered authentication packages is obtained by calling into the Local Security Authentication Subsystem (*Lsass.exe*). After a user's credentials have been authenticated, the logon process continues. There are two standard authentication packages used by Windows 2000 and Windows XP: MSV1_0 and Kerberos. The

MSV1_0 package is used to validate a user signing on to a standalone machine, and Kerberos is used to validate a user logging on to a domain.

The authentication of a user on a domain is similar to the authentication on a standalone machine. In this case, however, communication is across a network to the domain controller, necessitating a different authentication package for this operation. The security database for a domain is kept in Active Directory. The logon credentials are validated on the domain controller and then the account information is returned to the machine where the logon process is being performed. After the logon credentials have been validated, it is necessary to assign the logon session an access token.

Creating the Access Token

After a user's credentials have been authenticated, the job of creating an access token is the responsibility of the Local Security Authentication Subsystem (Lsass). The Lsass looks in the local policy database for the user's allowed access and, if the requested access is allowed, an access token is created. This access token is a data structure containing information describing the user's privileges and access rights. This particular token is the primary access token. After the access token has been created, it is passed back to Winlogon. Winlogon completes the logon process by running *Explorer.exe* and attaching the access token to this process. The *Explorer.exe* process now becomes the input desktop in the interactive window station and every process launched from the desktop is passed this access token. The primary access token is passed to the Win32 subsystem every time a process is launched, regardless of the mechanism used, and this token is attached to the process. The access token determines what any process that is launched can accomplish on the system.

Privileges and Access Rights

A privilege is an account right permitting a user to perform system-wide tasks. Examples of privileges are such things as being able to log on as a service, shut down the system, and change the system time. An administrator grants a privilege to a user account or a group account and this defines the access to system resources that this user or members of the group will have. An interesting point about privileges is that they have to be assigned, and many privileges are by default not assigned to anyone (even to an administrator). However, an administrator can assign those privileges that are by default not assigned to any user or group. Table 1-2 shows the default

privileges that are assigned to the various built-in accounts created on Windows 2000 Professional and Windows XP Professional.

Table 1-2: *Default Privileges for Built-in Accounts.*

Privilege	Account Names	Description
Access this computer from the network	Everyone, Users, Power Users, Backup Operators, Administrators	Allows a user to connect to the computer over the network.
Act as part of the operating system	<i>None</i>	Allows a process to authenticate as any user, and gain access to resources under any user identity. The LocalSystem account already includes this privilege.
Add workstations to domain	<i>None</i>	Allows the user to add a computer to a specific domain. When the user specifies the domain, an object is created in the Computer container of Active Directory on the domain controller.
Back up files and directories	Backup Operators, Administrators	Allows the user to override file and directory permissions in order to back up the system.
Bypass traverse checking	Everyone, Users, Power Users, Backup Operators, Administrators	Allows the user to navigate through directories to which the user otherwise has no access. This privilege applies to both the Windows file system and the registry. This privilege does not allow the user to list the contents of a directory, only to traverse directories.

(Continued)

Table 1-2: *(Continued)*

Privilege	Account Names	Description
Change the system time	Power Users, Administrators	Allows the user to set the time and date for the computer's internal clock.
Create a pagefile	Administrators	Allows the user to create and change the size of a pagefile.
Create a token object	<i>None</i>	Allows a process, run under the user account, to create a token, which it can then use to get access to any resources on the local computer. This privilege is already available with the LocalSystem account.
Create permanent shared objects	<i>None</i>	Allows a process, run under the user account, to create a directory object in the Windows 2000 object manager.
Debug programs	Administrators	Allows the user to attach a debugger to any process.
Deny access to this computer from the network	<i>None</i>	Denies a user the ability to connect to the computer over the network.
Deny logon as a batch job	<i>None</i>	Denies a user the ability to log on using a batch-queue facility such as the Task Scheduler.

(Continued)

Table 1-2: *(Continued)*

Privilege	Account Names	Description
Deny logon as a service	<i>None</i>	Denies a user the ability to log on as a service, as a means of establishing a security context. The LocalSystem account always has the right to log on as a service.
Deny logon locally	<i>None</i>	Denies a user the ability to log on at the local computer.
Enable computer and user accounts to be trusted for delegation	<i>None</i>	Allows the user to set the Trusted for Delegation setting on a user or computer object. A server process either running on a computer that is trusted for delegation or run by a user who is trusted for delegation can access resources on another computer
Force shutdown from a remote system	Administrators	Allows a user to shut down a computer from a remote location on the network.
Generate security audits	<i>None</i>	Allows a process, running under the user account, to make entries in the security log for object access auditing. As described later in this chapter, the SACL is used to define which types of access get audited.
Increase quotas	Administrators	Allows a process, running under the user account, access to another process to increase the processor quota assigned to the other process.

(Continued)

Table 1-2: *(Continued)*

Privilege	Account Names	Description
Increase scheduling priority	Administrators	Allows a process to have access to another process to increase the execution priority of the other process.
Load and unload device drivers	Administrators	Allows a user to install and uninstall Plug and Play device drivers. Device drivers that are not Plug and Play are not affected by this privilege and can only be installed by members of the Administrators group.
Lock pages in memory	<i>None</i>	Allows a process, running under the user account, to keep data in physical memory. This prevents the system from paging the data to virtual memory on disk. This privilege is considered obsolete.
Log on as a batch job	<i>None</i>	Allows a user to log on using a batch-queue facility such as the Task Scheduler.
Log on as a service	<i>None</i>	Allows a user account to log on as a service. The LocalSystem account always retains the right to log on as a service. Any service that runs under a separate account must be granted this right.

(Continued)

Table 1-2: *(Continued)*

Privilege	Account Names	Description
Log on locally	Guest, Users, Power Users, Backup Operators, Administrators	Allows a user to log on at the local computer.
Remove computer from docking station	Power Users, Administrators	Allows a user to undock a portable computer running the Windows 2000 or Windows XP operating system.
Replace a process level token	<i>None</i>	Allows a process, running under the user account, to replace the default token associated with a sub-process that has been started.
Restore files and directories	Backup Operators, Administrators	Allows a user to circumvent file and directory permissions when restoring backed up files and directories. It is also possible to set any valid security principal as the owner of an object.
Shut down the system	Users, Power Users, Backup Operators, Administrators	Allows the user to shut down the local computer.
Synchronize directory service data	<i>None</i>	Enables the user to read all objects and properties in Active Directory, regardless of the protection on the objects and properties.
Take ownership of files or other objects	Administrators	Allows a user to take ownership of any securable object in the system such as files, folders, and registry keys.

Access rights or permissions define who can access securable objects, and this access is granted or denied by the system based on the rights or permissions defined in the object. Access rights are set on an object-by-object basis and are defined in the object by the security descriptor that is always part of a securable object. Access rights are discussed in more detail later in this chapter. The important thing to remember is that privileges and access rights are different and that, when considering the issue of software installation, you need to be concerned with access rights.

Security Contexts

There are two security contexts that you need to know about if you want to understand how Windows Installer is able to install an application even if the user running the installation does not have administrative privileges. First, there is the user security context that is created for the logged-on interactive user. Second, there is the security context associated with the LocalSystem account. The OS subsystems and most Windows services run in the security context of the LocalSystem account.

User Context

Access to objects in the system depends on the rights defined in the access token associated with an interactive user's logon session. This access token is passed to the Win32 subsystem after a successful logon session and is associated with every process the user launches. This access token defines the *user security context*. Every time a process tries to access a securable object in the system, the access rights defined in the access token are compared with an Access Control List (ACL) maintained by the object. If the access rights of the user context are recognized by the object's ACL, the user is able to perform actions on the object. If the access rights are not recognized by the object's ACL, the user is denied access to the object. When it comes to software installation, the securable objects of greatest interest to you are files, folders, and registry keys.

After you log on to the system, the access token created when the logon session is successful can be thought of as your identity as far as Windows security is concerned. The access token is a securable object containing a number of items of information. However, only some of these items are of major interest and they are described in the following list:

User SID: The security identifier of the user account that is the owner of the current interactive logon session.

Group SID: A list of group security identifiers to which the logged-on user belongs. This includes the SIDs of built-in groups such as Everyone and Authenticated Users.

Privileges: A list of locally unique identifiers (LUID) that represent all of the privileges held by the owner of the User SID. These privileges can be directly assigned or can be those that are indirectly assigned by virtue of the user being a member of a group account.

Owner SID: The default security identifier that will be used to identify the owner of an object created during the logon session. Programmatically it is possible to specify a different owner SID when the new object is created.

Primary Group SID: The default group security identifier that will be used to identify the owner of any new object created during the logon session. This is used only for POSIX support.

DACL: The default discretionary access control list (DACL) that will be included in any new object created during the logon session.

The access token defines what operations you can perform on the system after you have logged on. Access to objects on the system is completely dependent on the privileges defined in your access token.

The LocalSystem Account

As discussed in this chapter, the LocalSystem account is the account that has no credentials and is attached to the system logon session. This means that it is present even if there is no interactive logon user. All processes running under the LocalSystem account run as part of the operating system. As you might imagine, this gives these processes a great deal of power to perform operations. In fact, the LocalSystem account has direct or indirect access to everything on the computer. Most Windows services are configured to run under this account.

In a network where the domain controller is Windows NT 4.0, the LocalSystem account on a computer has unlimited access on the local machine and essentially no access to anything on the network. With Windows NT 4.0, there is no mechanism to grant privileges to a computer. Where the domain controller is Windows 2000 server running Active Directory, computers on the network are now one of the types of objects that are stored in Active Directory. This makes it possible for a process running under the LocalSystem account to have access to other machines on the network. This access is possible only if a machine in the domain is granted privileges by the domain administrator.

Impersonation

Impersonation is the ability of an execution thread to execute in a security context different from that of the process owning the thread. A server thread uses an access token representing a client's credentials and, in so doing, the server can access only the resources that the client can access. When a thread of execution no longer needs to impersonate a different security context, it can revert back to its original security context and start using the access token of the process that launched it.

A good example of the use of impersonation is when a client process launched in the security context of the logged-on user launches a Windows service running under the LocalSystem account to perform a particular operation. The Windows service impersonates the security context of the user before it attempts to perform any operations. This impersonation is implemented to prevent the client process from carrying out any operations that are not possible under the security context of the client. If the user's access token defines the correct privileges, the operation succeeds; otherwise it fails. In the reverse of this last example, the Windows service may determine that it should not impersonate the security context of the client process requesting the service. The Windows service can then perform the operation under the normal privileges of the LocalSystem account. In this case, the operation succeeds because the LocalSystem account has complete access to everything on the computer. These two examples give you an idea of how Windows Installer provides an installation with *elevated privileges*. These are discussed in the next section.

Elevated Privileges

On Windows 2000 and Windows XP, Windows Installer runs as a client process and as a service process. It is the service process that runs the actions in the execute sequence table. The actions that make changes to the target system are written into an execution script and only these actions can receive elevated privileges.

An installation package using Windows Installer does not automatically receive elevated privileges when the installation is run. Elevated privileges have to be granted by the LAN administrator through the use of a Group Policy Object (GPO) or through the setting of a special registry key under both the HKLM and HKCU hives. Of course, if you have administrative privileges on your local machine, you already have the privileges necessary to install any application.

When a user has only user privileges on their local machine, they are unable to install software on that machine without elevated privileges. The normal mechanism that the Windows Installer uses to run an installation is to impersonate the logged-on user. Impersonation allows the service process of the Windows Installer to have only the same privileges that the logged-on user has. Now if elevated privileges are granted, the service process of the Windows Installer will no longer impersonate the logged-on user and will run in the security context of the LocalSystem account. Under this account the Windows Installer has access to all areas of the local computer.

If you create custom actions that make changes to the target system, you need to configure them properly so that they will have elevated privileges when such privileges are granted. A custom action will not have elevated privileges automatically, and this is discussed in detail in the chapters in Part II of the book.

In summary, elevated privileges are when the service process of the Windows Installer runs in the security context of the LocalSystem account and therefore has access to all areas of the file system and the registry. If elevated privileges are not granted, the service process of the Windows Installer impersonates the user and has only those privileges that have been granted to that user.

Access Rights

Access rights, also known as permissions, refer to the actions a user can or cannot perform on a securable object. Securable objects include files, folders, registry keys, Windows services, printers, and processes. Applications that create these securable objects set the security attributes for each object created. The Windows 2000 and Windows XP operating systems manage the security of an object by determining what user can access the object.

When a securable object is created, it is passed a data structure defining the object's owner and the operations a system user can perform on the object. This data structure is called a security descriptor and contains the SID of the user that created the object, a list of all the access rights for the object, a list of all the types of access that should be audited and written to the event log, and so forth.

When a process, such as *explorer.exe*, attempts to access a securable object, the object manager calls the Security Reference Monitor (SRM) to check whether the security descriptor in the object permits the type of access being requested. The SRM is the kernel-mode component of the Windows 2000 or Windows XP security subsystem. If the requested access is permitted, everything proceeds as normal. If the requested access is denied, the system displays an access denied message box.

The Discretionary Access Control List (DACL)

Other than the SID of the owner or creator of a securable object, the list of permissions that are allowed or denied for any particular user is the most important entry in an object's security descriptor. This list of permissions is called the discretionary access control list (DACL), and comprises a number of entries called Access Control Entries (ACEs).

Each ACE in a DACL contains three items of information. These are the SID of the user or group to which the ACE applies, a mask that defines the access rights that are being defined by the ACE, and a flag signifying whether the access rights are enabled or denied. As was mentioned when discussing the contents of a user's access token, one item of information in the token is a default DACL. The purpose of this default

DACL is to be inserted into the security descriptor for any object that is created without a specific security descriptor being defined. In fact, not defining a specific security descriptor is the standard approach used by many programmers when their application creates a securable object.

The System Access Control List (SACL)

The System Access Control List (SACL) is an Access Control List with another set of ACEs—except that there is a different purpose to these ACEs. The ACEs in a SACL specify those events that are to be reported in the event log for the particular user that tries to access the object. The SACL has no role in determining whether a user can access a particular object. There are two conditions under which an entry can be written to the event log: when an access attempt succeeds and when an access attempt fails.

Auditing can consume significant system resources and is therefore disabled by default in Windows 2000 and Windows XP. Because of this, a user must have administrative privileges to enable auditing.

Summary

This chapter explains how Windows Installer manipulates Windows security in order to provide elevated privileges to an installation package. Understanding elevated privileges is important when you want to create custom actions that will make changes to the target system. Windows security is also important with regard to installing an application to a locked-down system. Having a good grasp of the material in this chapter is important to understanding the material in a number of the chapters in this book.