Chapter 6: Understanding Network Access Security and Kerberos

Every organization has valuable resources it must protect from theft, vandalism, and even innocent clumsiness. Whether those resources take the form of cartons in a warehouse or files on a hard drive, the fundamental aspects of security remain the same:

- **Authentication.** Individuals must be verified to have authorized access to controlled areas.

- **Access control.** All possible entry points must be blocked and guarded. Sensitive areas may need additional internal access controls.

- **Audit.** Access must be monitored and responsible personnel informed immediately if an access infraction occurs.

This chapter covers the authentication and audit features available in Windows 2000. Access control is covered in Chapter 14, "Managing File Systems Security," and Chapter 15, "Managing Shared Resources." Additional authentication and access control information pertaining specifically to remote access is covered in Chapter 17, "Managing Remote Access and Internet Routing."

Access Security Overview

Since its inception, classic NT has used a proprietary authentication scheme called **NT LAN Manager (NTLM) Challenge-Response.** With Windows 2000, Microsoft adopted a public domain authentication scheme called **Kerberos.** Kerberos was developed at MIT as part of Project Athena. It takes its name from the mythological three-headed hound that guarded the gates of the underworld in Roman mythology. (If you’re a humanities scholar making the transition to Information Technology, you may wonder why Project Athena picked a Roman mythic creature rather than the Greek counterpart, Cerberus. I can’t help you. Computers and classics just don’t mix.)


Because the authentication mechanism is designed to be as transparent as possible, it isn’t all that obvious that Kerberos is at work rather than the classic NTLM Challenge-Response. Windows 2000 uses Kerberos in the following circumstances:

- Authenticating users logging on to Windows 2000 domain controllers

- Authenticating users logging on to Windows 2000 servers and workstations that are members of a Windows 2000 domain

- Authenticating users logging on to standalone Windows 2000 servers and workstations
Authenticating users accessing a Windows 2000 server or workstation from a Windows 9x client configured with the Active Directory add-on

NTLM Challenge-Response authentication is used in the following instances:

- Authenticating users logging on to Windows 2000 servers and workstations that are members of a classic NT domain (or accessing a classic NT domain from a Windows 2000 domain via a trust relationship)
- Authenticating users accessing a Windows 2000 server or workstation from a classic NT server or workstation
- Authenticating users accessing a Windows 2000 server from a standard Windows 9x or 3.1x client

If you find yourself wondering how to verify this, you can enable auditing and examine the logged transactions, because a user logs on both at the console of a member workstation and the console of the server. See the "Auditing" section later in this chapter.

**Functional Description of NT Security Architecture**

It’s difficult to point your finger at one place in the Windows 2000 architectural model and say, "Here is where you find security-handling services." Security is integrated into every aspect of the operating system. The majority of security decisions are controlled by the **Local Security Authority (LSA)**. The LSA calls upon logon services, such as WINLOGON and NETLOGON, to obtain security credentials from users. After it has obtained the user’s credentials, the LSA performs its authentication chores with the help of **security providers**. After a user has been authenticated, he or she receives an access token that identifies all processes initiated by the user. The token identifies the user’s security code along with any groups and special privileges associated with that user.

Microsoft makes it possible for third-party vendors to extend and modify the base authentication mechanism for console logons. Normally, the user provides credentials in the form of a name/password combination, but other packages use thumbprints, voiceprints, retinal scans, answers to nosey questions, and someday probably even invasive surgery. Companies go to great lengths to keep unauthorized users from running Solitaire on their networks. These third-party packages can also work by relying on Microsoft’s standard authentication engine with modifications to the credential-handling mechanism. This is done using a special library of security routines that make calls to the **Graphical Identification and Authentication (GINA)** library. A common example is the NWGINA used by the NetWare client for Windows 2000.

**Local Security Authority (LSA)**

The LSA uses a typical NT/Windows 2000 client/server subsystem arrangement consisting of a User-mode portion running in Ring 3 and an Executive portion running in Ring 0. The User-mode side consists of the **Local Security Authority SubSystem**, or LSASS.EXE. The Executive side consists of the **Security Reference Monitor** (SRM). The LSASS includes two services that collect user credentials: WINLOGON and NETLOGON, and a set of security providers, part of the SSPI, that
process these credentials to verify that the user is authorized to access the computer or domain or both. In the Registry, these security providers are called *security packages*.

**Security Providers and the SSPI**

The SSPI provides a configurable and flexible way for Windows 2000 to interact with security systems. The SSPI enables programmers to use a single set of API calls to handle authentication chores instead of forcing them to make provisions for any and all types of local, network, Internet, public/private key, and proprietary authentication mechanisms. SSPI provides the same flexibility for authentication systems that Network Device Interface Specification (NDIS) provides for network systems and Open Database Connectivity (ODBC) provides for database management systems.

Security providers take the form of DLL libraries that snap into the LSASS executable. Third-party vendors who want to develop new and nifty security packages can write their own providers. As Windows 2000 matures, look for more and more third-party SSPI providers as vendors look for ways to leverage this new feature to their advantage. The following provider packages come with Windows 2000:

- **Kerberos (KERBEROS.DLL)**. This provider supports Kerberos clients. When a Windows 2000 client attempts to access a Windows 2000 server, the client calls on KERBEROS.DLL to handle the authentication. As discussed in the next topic, the server side of this Kerberos transaction is controlled by the Kerberos Key Distribution Center service, KDCSVC.DLL, running on a Windows 2000 domain controller.

- **NTLM Challenge-Response (MSV1_0.DLL)**. This provider supports classic NTLM Challenge-Response authentication. A complementary implementation of NTLM Challenge-Response for Internet services is provided by WINSSPI.DLL. This provider supports WWW services that use WINSSPI to authenticate users who initiate server-side scripts running CGI, ActiveX, or Windows Script Host.

- **LSA Negotiate (LSASRV.DLL)**. This provider interacts with WINLOGON and NETLOGON to pass security credentials to the security providers.

- **Distributed Password Authentication (MSAPSSPC.DLL)**. The DPA provider supports the Microsoft Network (MSN) and other large content providers.

- **MSN authentication (MSNSSPC.DLL)**. This provider supports an older, proprietary authentication used by MSN before DPA.

- **Secure Socket Layer/Private Communications Transport (SSL/PCT) (SCHANNEL.DLL)**. This provider supports secure Internet communications. For example, this provider is used when Internet Explorer secure API calls are made to WININET.

- **Digest authentication (DIGEST.DLL)**. This provider supports a new method for authenticating WWW users. It is an extension of the standard basic Web authentication, but does not require transfer of the user’s password.
Registry Tip: Security support providers

The list of security support providers is located in HKLM | System |
CurrentControlSet | Control | SecurityProviders.

Control parameters for the LSA and its security support providers are contained in HKLM |
System | CurrentControlSet | Control | LSA.

Account and Security Databases

The various security support providers need a way to store credentials for users, groups, and computers. Classic NT and standalone Windows 2000 computers store security information in three Registry-based security databases. Windows 2000 domain controllers store security information in the Active Directory.

The three classic NT security databases are as follows:

- **Builtin.** This database contains the two default user accounts, Administrator and Guest, along with various default groups such as Domain Users for domains and Power User for workstations and standalone servers. The Builtin database is part of the SAM Registry hive in the HKEY_Local_Machine (HKLM) subtree. This and other Registry hives (except for user profiles) are located in the Winnt\System32\Config directory. The structure of the Builtin database differs between standalone servers and domain controllers. This is one of the reasons that a classic NT server requires reinstallation to upgrade it to a domain controller. A Windows 2000 domain controller migrates the Builtin accounts to the Active Directory when it is promoted.

- **Security Account Manager (SAM).** This database contains classic NT user and group accounts created after the initial installation of Windows 2000. This database is contained in the SAM Registry hive. Each user, group, and computer is assigned a security ID (SID). The Security Reference Monitor uses SIDs to control access to security objects such as files, Registry keys, and Active Directory objects. See the "Security ID Codes" section later in the chapter for details on SID construction.

- **LSA.** This database contains the password rules, system policies, and trust accounts for the computer. It is kept in the SECURITY Registry hive, also under the HKEY_Local_Machine subtree. The SECURITY hive also contains a copy of the SAM database.

Peeking Inside the Hives

Ordinarily, you cannot view the contents of the three classic security databases because the Registry keys permit full access by the System account only. If you want to take a peek inside these hives, you can set the permissions on the Registry keys to give your account or the Administrators group account full control access. Do not do this on a production computer. You will not necessarily wreck the security database, but you take
a big chance of corrupting an entry. All data in the security databases is encrypted and stored in binary format. Refer to Figure 6.1 for an example of the hive structure.

---

Figure 6.1
SAM database viewed by the Registry Editor after changing security permissions.

Computer Accounts

The SAM database contains accounts for computers as well as users and groups. Computer accounts create miniature trust relationships with a domain controller. These trusts are used to establish secure communications links using MS Remote Procedure Call (MSRPC), which the local LSA uses to pass through the user’s authentication request to a domain controller. The Windows 2000 computer account prevents someone from plugging an unauthorized computer on to the network and getting access to the domain.

Every computer account has a password associated with it that is generated when the computer joins the domain. You can’t see this password from the UI. It changes every 28 days via a secure negotiation between the computer and a domain controller.

---

Expired Password

It sometimes happens that a user cannot connect to the domain after an extended absence because the local computer account password has expired. If this happens, you must take the workstation out of the domain, delete the computer’s domain registration, and then rejoin it to the domain. You must also re-register a workstation if you change its name. This gives the computer a new domain SID. If you attempt to join a computer to a domain and give it the same name as an existing computer, the domain controller will reject the registration request, even if the existing computer is no longer on the network. Be sure to delete old computer registrations.

---

Passwords

Neither the SAM nor the Active Directory stores a clear-text copy of a user’s password. Instead, the password is encrypted using RSA MD4 encryption. The MD4 algorithm takes a variable-length password and encrypts it using a secret key to produce a fixed-length result called a message digest. The message digest is a one-way function, meaning that it cannot be decrypted back to its original form. This one-way function is often called a hash function, or sometimes just a hash. This is because MD encryption permits mixing together discrete elements when forming the message digest.

The more bits there are in the message digest, the harder it is to break. Domestic versions of Windows 2000 use a 128-bit message digest, considered unbreakable by anyone but the National Security Administration (NSA), and only then with a great deal of expensive supercomputer time. Windows 2000 increases the complexity of the message digest by using case-sensitive Unicode characters. Export versions of Windows 2000 are limited to a 40-bit message digest, considered easily crackable by folks whose job it is to do such deeds.
The security picture is much bleaker when you look at the support for down-level clients, such as Windows 9x and Win31. Older versions of Windows use a LAN Manager–compatible password encrypted using the U.S. Data Encryption Standard (DES). Not only is DES encryption not nearly as secure as MD4, the LAN Manager password itself is more easily determined because it is not case sensitive and uses only ANSI characters. A number of tools have bubbled up in the Internet that can scan a classic SAM database, nab LAN Manager passwords, and use them to open a system. Starting with Windows NT 4 SP3 and continuing into Windows 2000, the SAM can be protected with special encryption to protect these LAN Manager passwords. This requires generating a special system key that is used to encrypt the password store. As of this writing, no one has successfully cracked the SAM or the Active Directory on a Windows 2000 computer. Still, you should carefully guard access to the security databases just in case some bright 14-year-old devises a hack. This includes protecting backup tapes and anything else that holds backup copies of the Registry, such as RDISK or REGBACK. Network access to the server does not expose the SAM or the directory because they are locked on a running server. A potential backdoor exists because the RDISK utility saves an unlocked copy of the SAM in the WINNT\Repair directory.

LAN Manager passwords also represent a security hole because they are transmitted over the wire during the authentication process. Even if you do not have down-level clients, this makes it possible for an unauthorized person to collect the DES-encrypted passwords. You can prevent a Windows 2000 server from challenging for a LAN Manager password, and disable Windows 2000 clients from sending them, so long as you do not have down-level Windows 9x or 3.1x clients in your network. Use the following Registry entry to disable LAN Manager passwords:

```
Key:    HKLM \System\CurrentControlSet\Control\LSA
Value:  LMCompatibilityLevel
Data:   2 (REG_DWORD)
```

**Security ID Codes**

All Windows 2000 and NT computers, both servers and workstations, get a unique SID during initial setup. Standalone servers and workstations use the local computer SID to build SIDs for users and groups stored in the local SAM. Domain controllers use the SID assigned to the domain SID for building user, group, and computer account SIDs.

A SID consists of a series of 48-bit (6-byte) components that identify the issuing authority, the identity of any sub-authorities that determine how an account bearing this SID can interact with the operating system, and a random number that uniquely identifies the computer or domain. The binary content of the SID is rendered into alphanumeric format when exposed to the user interface. Here is an example of a computer SID:

```
S-1-5-21-1683771067-1221355100-624655392
```

The bold portion uses the format: **S-R-IA-SA**

- **S** represents the **SID Identifier**. It flags the number as a SID rather than some other kind of long, obscure number.

- **R** represents the **Revision**. All SIDs generated by Windows 2000 and classic NT have a revision level of 1.
• **IA** represents the **Issuing Authority**. Nearly all SIDs use the NT AUTHORITY as the issuing authority. The NT AUTHORITY designator is 5.

• **SA** represents one or more the **Sub-Authorities**. These authorities designate special groups or functions.

A user’s SID consists of the SID of the computer or domain holding the user’s account, followed by a sequential number called a relative ID (RID). Here is an example of a user SID. The bold portion is the RID:

\[
\text{S-1-5-21-1683771067-1221355100-624655392-500}
\]

Every user, computer, and group in a domain is given a RID by the primary domain controller (PDC) when the account is created. The default Administrator account gets RID 500 and the Guest account gets RID 501. RIDs for other users, groups, and computers are doled out in sequence on a first-come, first-served basis starting with decimal 1000, or hex200. The SAM stores user and computer accounts in the same database. It differentiates between them by appending a dollar sign to the computer names. A registered domain member computer with the name PHX-W2K-003, for example, would have a computer account name in the SAM of PHX-W2K-003$. In the Active Directory, user accounts and computer accounts are assigned different object classes that have different attributes, but even then the classes are virtually identical.

If you’re new to NT/Windows 2000, this business of SIDs and RIDs might seem like geek-level stuff that no one really cares about except design engineers. Nothing could be further from the truth. Understanding how SIDs are generated, stored, and manipulated is absolutely vital to troubleshooting otherwise inexplicable security events. Just like Deep Throat telling Bob Woodward to follow the money, when you troubleshoot Windows 2000 security problems, you must follow the SIDs. What if Nancy Atchison in Accounting marries Bill Topeka in Engineering, for example, and they decide to hyphenate their last names and insist that you change their logon names? When you change natchison to natchisontopeka and btopeka to batchisontopeka, the change has no impact on their user rights because the underlying SIDs remain the same. If, on the other hand, you delete their accounts and built new ones with their new names, they would lose their access rights and group memberships. Not a pleasant prospect when you consider that these two users might be members of dozens and dozens of groups in addition to having their individual accounts included on file and directory access lists in resource domains all over the world.

**SIDs**

Certain SIDs represent standard local and global groups. These are called **well-known SIDs**. The groups represented by these SIDs have special significance to the operating system. For example, a special local group called **Interactive** has a SID of S-1-5-4. Any user who logs on at the console of a Windows 2000 computer is made a member of the Interactive group and is given whatever rights have been assigned to that group. On Windows 2000 Professional desktops, the Interactive group is a member of the Power Users group, giving a local user even more system rights.

Some well-known SIDs represent accounts rather than groups. For example, SID S-1-5-18 represents the Local System account, which provides a security context for background services. This Local System account is important to keep in mind because it comes into play when processes running on
one computer need to access resources on another computer. Table 6.1 shows the well-known SIDs and their functions

### Table 6.1 Well-Known SIDs and Their Functions

<table>
<thead>
<tr>
<th>SID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-1-0-0</td>
<td>A memberless group used to represent an account with no known SID. This is also called a <strong>null SID</strong>.</td>
</tr>
<tr>
<td>S-1-1-0</td>
<td>This is the World SID, which is known in Windows 2000 and NT as the <em>Everyone</em> group. This group includes all users who log on to the domain, including anonymous users accessing resources from the Internet.</td>
</tr>
<tr>
<td>S-1-2-0</td>
<td>The <em>Local</em> group. This group includes users who have physically logged on to the console of the computer.</td>
</tr>
<tr>
<td>S-1-3-0</td>
<td>The <em>Creator/Owner</em> group. This group designates the user or service that created a security object or currently owns a security object. This number acts as a placeholder and is normally not displayed in the user interface.</td>
</tr>
<tr>
<td>S-1-3-1</td>
<td>A special form of the <em>Creator/Owner</em> group that contains the primary group for an account. Windows 2000 uses the primary group to support Macintosh users in a domain. The default primary group for users in a local SAM is Users. The default primary group for users in the Active Directory is <em>Domain Users</em>. The primary group can be changed, if necessary.</td>
</tr>
<tr>
<td>S-1-5</td>
<td>The <em>NT AUTHORITY</em>. SIDs for special groups issued by this authority start with S-1-5</td>
</tr>
<tr>
<td>S-1-5-1</td>
<td>Dial-up</td>
</tr>
<tr>
<td>S-1-5-2</td>
<td>Network</td>
</tr>
<tr>
<td>S-1-5-3</td>
<td>Batch</td>
</tr>
<tr>
<td>S-1-5-4</td>
<td>Interactive</td>
</tr>
<tr>
<td>S-1-5-5</td>
<td>Logon used to control process access between sessions</td>
</tr>
<tr>
<td>S-1-5-6</td>
<td>Service</td>
</tr>
<tr>
<td>S-1-5-7</td>
<td>Anonymous logon</td>
</tr>
<tr>
<td>S-1-5-8</td>
<td>Proxy</td>
</tr>
<tr>
<td>S-1-5-9</td>
<td>Server logon (also called the Domain Controllers or Enterprise Controllers account)</td>
</tr>
<tr>
<td>S-1-5-10</td>
<td>Self</td>
</tr>
<tr>
<td>S-1-5-11</td>
<td>Authenticated user (added in NT4 SP3, to differentiate S-1-1-0 <em>Everyone</em> users who have received network authentication from S-1-1-0 <em>Everyone</em> users who are touching the network</td>
</tr>
</tbody>
</table>
RIDs

Maintaining a sequential list of RIDs is automatic in classic NT because only the PDC can add new users, groups, and computers to the domain SAM database. The situation gets a bit more complicated in a Windows 2000 domain because every domain controller can write to the Active Directory.

A Windows 2000 domain has a single RID pool that is passed around among domain controllers like kids sharing a Tootsie Roll. Each domain controller carves off a chunk of 100,000 RIDs from the RID pool and uses them when it needs to add a new user, group, or computer to the Active Directory. This means that a Windows 2000 domain could have nonsequential RIDs. Classic Backup Domain Controllers (BDCs) will refuse to replicate if they see that the RIDs do not conform to standards.

To provide backward compatibility for classic BDCs, a Windows 2000 domain has a special operating configuration called Mixed mode. In a Mixed-mode domain, one Windows 2000 domain controller is designated as the RID master and retains exclusive use of the RID pool. By default, the RID master is the upgraded PDC. This server also becomes a PDC master to support the classic domain requirement that replication must come from a single server. The PDC master and RID master do not necessarily need to be the same server, but that configuration makes the most sense. The PDC master and RID master can be transferred to another Windows 2000 domain controller. From the perspective of the classic BDCs on the network, this appears to be a promotion of a BDC to a PDC. They acknowledge the change and begin replicating from the new domain controller.

When all classic NT domain controllers have been upgraded or retired so that every domain controller is running Windows 2000, the network can be shifted to Native mode. This is a one-time event and cannot be reversed. No classic NT domain controllers can ever again be introduced into a Native mode domain. The RID master releases the RID pool to other Windows 2000 domain controllers and the PDC master becomes irrelevant.

Access Tokens

After a security provider authenticates a user, LSASS calls on the Security Reference Monitor (SRM) in the Windows 2000 Executive to assemble the user’s access token. The access token accompanies the threads of all processes spawned by the user. The token contains the user’s SID along with the SID of any groups to which the user belongs. If a user logs on at the console of a standalone Windows 2000 Professional desktop, for example, the user gets an access token containing the user’s
SID, the S-1-2 local group, the S-1-5-4 Interactive group, and the SID for the Power Users group to which the Interactive group belongs. Each of these SIDs has certain defined permissions bestowed on the user because they are in the access token. The token also contains security restrictions that apply to the user, such as logon hours and password expiration.

An access token does not accompany the user across the network. Access permission, like politics, is always local. When a user attempts to connect to a server, or initiates a process that attempts a connection, the local security authority first authenticates the user using processes described in this chapter, and then impersonates the user to obtain an access token that accompanies any processes initiated by the user.

**Limitations of Classic NT Security**

Classic NT Registry-based security has the following eight major limitations:

- Restricted SAM size
- Multiple logon IDs
- PDC represents single point of failure
- Poor operational performance
- Poor replication performance
- Lack of management granularity
- Security databases differ between servers and domain controllers
- Nontransitive trust relationships

**Restricted SAM Size**

The total number of users, computers, and groups in classic NT is limited because the SAM cannot grow much above 80MB or so. This is due to restrictions on overall Registry size, called the *Registry size limit* (RSL), which keeps the Registry from consuming all the available paged pool memory. Generally speaking, paged pool memory represents the total available RAM less about 4MB set aside for the non-paged pool memory used by the Executive. You can view the settings for the various memory pools in `HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management`. The default values for these memory pools are zero, indicating that they are calculated dynamically. You should not change any values unless you have specific directions from Microsoft Technical Support.

One setting that can be adjusted is the RSL. This is usually done automatically by the system when the Registry grows to the point that it is about to exceed the RSL. You can also adjust the RSL manually using the System applet in Control Panel. Select the Advanced tab, click Performance, and then click Change to open the Virtual Memory window. The value in the Maximum Registry Size field is the RSL. This is not the amount of memory actually set aside for the Registry, but the maximum size that the Registry is permitted to grow. The default RSL is 25% of paged pool memory.
The RSL is permitted to grow to 80% of paged pool memory with a ceiling of about 108MB. The SAM is only one component of the Registry, so it is restricted further to about 80MB. Each user account uses 1KB, with 0.5KB for each computer account and 4KB for each group account. A typical SAM has enough room for about 40,000 users if you count the groups you’ll need to manage them. In practical terms, poor replication performance and slow logon times cut the maximum number of users to about 15,000.

**Multiple Logon IDs**

In an ideal universe, the domain logon account would provide access to all server-based applications. In the real world, client/server application designers have been slow to use the flat-file security database in classic NT. This forces users to have separate logon IDs for the domain, email, host access, groupware, Internet proxies, and so forth.

**Single Point of Failure**

Updates to the classic NT security databases can only be done by contacting the PDC. If the PDC in the master domain crashes, users cannot change their passwords and you cannot add new users or domain global groups anywhere in the world. To correct this problem, an administrator must promote a BDC to PDC somewhere in the domain. If the promoted BDC doesn’t have the horsepower of the original PDC, worldwide performance suffers. A worse situation occurs if the WAN connection that connects the PDC to the rest of the domain goes down. In this situation, you don’t dare promote a BDC because when the WAN connection returns, you’ll have two PDCs with slightly different security database contents. This forces you to make a Solomon decision, keeping one PDC and killing the other. In short, you have the makings of a real disaster.

**Poor Operational Performance**

The single PDC in a classic NT domain also imposes practical limits on daily operations. Assume, for example, that you are an administrator of a global NT network with 30,000 users. You are stationed in Omaha, but the PDC for the master security domain is Boston. You open User Manager for Domains to add a new user. Depending on the speed of the WAN link, it can take a long, long time to read the huge, flat-file SAM database across the WAN from Boston to Omaha. Most administrators in large NT domains learn to use command-line utilities to avoid this irritation.

**Poor Replication Performance**

The hub-and-spoke replication model of classic NT imposes operational limits beyond the problem with limited SAM size. A large network with many BDCs imposes a great deal of load on the PDC to keep the databases replicated. By default, replication occurs when 200 updates accumulate or every seven minutes or at a random interval between one and seven minutes. If you don’t want to wait for replication to carry an update to a remote BDC, you must use Server Manager to force replication. This means opening still another tool and waiting another period of time.

**Security Databases Differ Between Servers and Domain Controllers**

The SAM database has a different structure on a domain controller than on a regular server. A classic NT server cannot be promoted directly to domain controller, and it cannot be demoted from a domain controller back to a server. The difference in the security database structure makes it necessary to
completely reinstall NT if you want to change the server’s security role.

Lack of Management Granularity

Administrators in one location in the domain have administrator privileges everywhere in the domain. Assume, for example, that a user forgets his password one morning and, after trying a few different entries, eventually picks up an intruder lockout. This happens all the time, right? Who can the user call for help? A local administrator? No. Local administrators have no rights in the master security domain. Only administrators in the master domain can perform password resets and intruder lockout resets and other routine daily services.

The central staff in a big network generally lift their noses at this kind of work, however, and try to delegate responsibility back to the local administrators. Unfortunately, there is no such thing in classic NT as a "Regional Domain Admin" or "Limited Domain Admin." There is an Account Operator group, but the management privileges of this group extend throughout the domain. This means an administrator hired and managed by the IT group in one city can make changes to accounts in another city. (Most IT managers I know tend to frown and get red in the face and make calls to their CIOs when they find out about this kind of situation.)

Several third-party tools have become popular to overcome this lack of management granularity in NT. Examples include Enterprise Administrator from Mission Critical Software, http://www.missioncritical.com/, and Trusted Enterprise Manager from MDD, Inc., http://www.mddinc.com/. These products filled a need in classic NT and they might find their way into the Windows 2000 market, but they carry their own replication and management baggage that might not be suitable for your environment. They are worth a look, however.

Nontransitive Trust Relationships

Several domains can be linked together to form trust relationships. In classic NT, however, these trusts come strictly in pairs. If Domain A trusts Domain B, and Domain B trusts Domain C, then Domain A does not trust Domain C or vice versa. Managing a large network with many interlocking trusts can turn highly competent administrators into slobbering maniacs. You know when you walk into the operations center of a big classic NT shop because there’s butcher paper on the walls with circles and arrows going everywhere.

Windows 2000 Kerberos Authentication

In light of all the functional and operational limitations in classic NT security, it was imperative that Microsoft improve the situation in Windows 2000. Rather than try to incrementally fix the classic NT security structure, Microsoft chose to adopt entirely different security methods while maintaining backward compatibility with classic NT. The LDAP-based Active Directory replaced the old flat-file Registry databases and, to replace the aging NTLM Challenge-Response authentication, Microsoft chose another open standard, Kerberos.

Kerberos, the mythological hound, had three heads so that it could guard all ways in and out of Hades. Kerberos, the modern authentication mechanism, uses the following three parties to validate authorized users:

- A user who is trying to gain access to resources on a target server
The target server that needs to validate the user’s identity before giving access

A special server that holds credentials for both the user and the target server so that it can perform the necessary authentications to authorize the user’s access

The three-way nature of Kerberos authentication resolves nearly all the authentication-related limitations in classic NT security. Kerberos supports fully transitive trusts between domains, making it possible for administrators in Domain A to use groups from Domain C via a trust to Domain B. This sets the stage for more elaborate domain trust configurations without additional management overhead. Kerberos also provides mutual authentication and periodic re-authentication, making it a more secure protocol than NTLM Challenge-Response. And best of all, Kerberos is much faster than NTLM.

The functional overview portion of this chapter covers how the three-way Kerberos authentication takes place. Kerberos has its own lexicon, so the overview also introduces a lot of new vocabulary. The remaining topics cover how to use the new security policies in Windows 2000.

**Kerberos Overview**

Kerberos transactions resemble a scene from a John Le Carre spy novel. Imagine that a mole needs to contact her parent spy organization. She sends a prearranged signal, and the parent organization agrees to send a runner to meet her. The mole has never seen the runner before. The runner has never seen the mole before. How does each know that the other is genuine? Simple. They have a common acquaintance, the Chief back at headquarters. Here’s how it works:

**Procedure 6.1 Kerberos Authentication Transaction**

1. The mole calls the Chief and says, "Give me a secret that only you and the runner know."

2. The Chief, always security-conscious, verifies that the mole is genuine by asking for a special signal known only to members of the secret service.

3. If the mole gives the correct sign, the Chief gets out the personnel files for the mole and the runner. A personnel file contains a secret encryption key known only to that spy.

4. The Chief then builds a message to the mole. The message has two parts:
   
   - Part one contains a random number thought up by the Chief and encrypted with the mole’s secret key.
   
   - Part two contains the same random number, the mole’s name, the time the Chief made the note, and a duration beyond which the note is not valid, all encrypted with the runner’s secret key.

Anyone reading this message would not glean any useful information from it. Only the Chief can read the whole thing, and he’s so absent-minded that he forgets the random number as soon as he gives the message to the mole. No one could torture him to obtain it.
5. The mole uses her secret key to decode the random number in her portion of the message. If the result is gibberish, she assumes that someone has impersonated the Chief and given her a fake message. If the number looks right, she puts the runner’s portion of the message in her purse for safekeeping.

6. That afternoon the mole and the runner meet. They exchange names. The mole gives the runner the second part of the Chief’s message. She watches him closely. This is a moment of truth.
   - If the runner cannot decode the Chief’s message, the mole knows the runner is bogus and shoots him.
   - If the runner can decode the message, but the contents inside are scrambled, the runner knows the mole has fiddled with the message and he shoots her.
   - If the runner decodes the message and the mole’s name in the message is different from the name she gave when they met, he shoots her.
   - If the runner has ever seen the random number before, he shoots her.
   - If the time of day exceeds the allowable duration of the Chief’s message, the runner throws away the message and walks on.

7. If none of these unfortunate circumstances occur, the mole hands the runner another message. The mole devised this message. It contains her name, the current time, and the total number of letters in the message. The mole coded this message using her copy of the Chief’s random number as an encryption key.

8. The runner uses the random number he obtained from his part of the Chief’s message to decode the mole’s message. If the result is garbled, one of them is an imposter, but they don’t know which one. The scene comes to a Quentin Tarantino ending: The mole shoots the runner and the runner shoots the mole.

9. If the runner can decode the mole’s message, the three-way authentication is complete and they begin sharing information.

I know extended metaphors make for slippery examples, but this game of *I Know A Secret, Do You?* follows a Kerberos authentication under Windows 2000 fairly closely. The actual transactions are a bit more complicated only because the principals can’t meet face to face, so to speak. They send out their messages over a public network and so must assume that a bogeyman is out there capturing packets and using them to infiltrate the network.

Before examining Kerberos transactions in detail, it’s important to understand the terms and expressions used by Kerberos. They differ significantly from those used by NTLM.

**Kerberos Vocabulary**

Kerberos has existed in the public domain for years and has a colorful language all its own. The mix of this terminology with those used to describe classic NT authentication makes for a hodge-podge of
lingo that is dense, even by network systems standards. The following list of Kerberos terms explains their meaning and maps them to their classic NT counterparts.

**Security Principal**

Any authentication mechanism must be able to validate entities—whether they are users or computers or other devices that desire access to local or network resources. The Kerberos term for these entities is *security principals*, or often just *principals*. Security principals in Windows 2000 are *users*, *groups*, and *computers*. Each of these security principals is assigned a SID for use in the Windows 2000 object-based security system.

**Realm**

Every authentication system—Kerberos is no exception—requires a database to hold credentials. A Kerberos *realm* is defined by the contents of the database that holds the credentials for its security principals. The terms *domain* and *realm* are synonymous in Windows 2000. All objects in a domain, including those representing security principals, are contained in a single Active Directory database.

**Ticket**

The *ticket* is the fundamental unit of currency in Kerberos transactions. The ticket contains encrypted information used by the three parties in a Kerberos transaction to authenticate a security principal. When a security principal attempts to access a server, a ticket for that server must be presented. How that ticket is obtained, submitted, and validated is covered in the "Analysis of Kerberos Transactions" section a little later in this chapter.

**Key Distribution Center**

The central service that distributes Kerberos tickets is called the *Key Distribution Center* (KDC). A *key* and a ticket represent the same concept, but the term *key* is rarely used in contexts other than the KDC. The KDC has two primary functions: an authentication service and a ticket-issuing service. Some Kerberos implementations use separate servers for these two services, but this is not a requirement. Windows 2000 uses a single Key Distribution service, KDCSVC, to both authenticate security principals and issue their tickets. The KDCSVC runs only on domain controllers. Other Windows 2000 computers and Windows 9x computers with the Active Directory add-on have client services that are used to communicate with the KDCSVC using Kerberos protocols.

Kerberos is an open protocol, so, theoretically, Windows 2000 clients can get Kerberos tickets from a KDC on any platform and vice versa. In practice, however, subtle differences between Kerberos implementations make interoperability a challenge. So far, Microsoft has stipulated compatibility only with MIT Kerberos 5.

**Ticket-Granting Service**

The Ticket-Granting service is one of the two functions of a KDC. Whenever a security principal reaches out across the network to touch a server, it must present a Kerberos ticket for that server. The security principal obtains this ticket from a Ticket-Granting service. In Windows 2000, the Ticket-Granting service is incorporated into the KDCSVC service on a domain controller. It is not listed separately on a process list.
A ticket authorizes a user to access only a specific server. If a Windows 2000 user maps a drive to a shared folder on five servers, for example, the Kerberos client service must contact the KDCSVC service on a domain controller and obtain tickets for each of those servers on behalf of the client. The ticket is submitted to the target server as part of the initial server message block (SMB) connect command.

Authentication Service

Authentication is the second function of a KDC. Before a user can obtain a ticket, he must be verified to be an authorized security principal in the realm of that KDC. The Authentication service performs this function by checking the security principal’s credentials against the contents of the security database. Windows 2000 uses the Active Directory for this database. When the KDCSVC on a Windows 2000 domain controller authenticates a user, it issues a ticket-granting ticket (TGT). The TGT speeds up the ticket-granting process by preauthorizing the user. In subsequent transactions when the user wants to access a specific server, the Kerberos client server submits the TGT to the KDC to quickly get a ticket for the target server. It’s like eating Thanksgiving dinner at your grandmother’s house. You must first get permission to eat at the family table. Only later do you get permission to dig in.

Validating Server

The validating server is the third party in the three-way Kerberos transaction. A Kerberos validating server is equivalent to a Windows 2000 member server. When a user attempts to access a member server, the Kerberos client submits a ticket for that server obtained from the KDC. The member server validates that ticket by checking the contents for encrypted messages that only an authorized security principal could possibly put there. A validating server must belong to the same Kerberos realm as the KDC that issues its tickets.

Transitive Trusts

Kerberos transactions in Windows 2000 are called transitive because KDCs in trusted domains work together transparently to point Kerberos clients at the proper domain controller to obtain tickets. A classic NT domain trust relationship could only be established between pairs of domains because the SAM was limited in what it could store about a trusted domain. Using Kerberos, the classic trust relationship can be expanded to include trusts between remote domains.

Assume, for example, that Domain B trusts Domain A and Domain C trusts Domain B. In classic NT, administrators from Domain C could not add users and groups from Domain A to their local access control lists. In Windows 2000, users and groups from any of the three domains can be added to groups and access control lists in any other domain.

The complexities of classic NT trust relationships have always caused migraines for administrators. Most big NT shops have multicolored trust interconnection diagrams hanging on the walls of their server rooms; the administrators use these when creating groups and assigning access rights. Kerberos transitive trusts won’t eliminate the need for those drawings; after you have fully migrated to Windows 2000, however, they should look more like engineering diagrams and less like Dr. Seuss illustrations.
KRBTGT Account

Kerberos uses a special identifier to differentiate ticket-granting tickets issued by KDCs in different realms. This identifier is a combination of the realm name and the password associated with a special account called krbtgt. The KDC uses a hash of the krbtgt password to encrypt a random number called a nonce that it includes in the data field of all TGTs. This nonce makes it difficult to hijack or copy authentication tickets because the krbtgt key is known only to the original domain.

The krbtgt account is built automatically when the first server in the domain is promoted to domain controller. The account cannot be deleted and the name cannot be changed. You can change the password, but this is not recommended. Changing the password has no adverse effect, per se. It does invalidate any outstanding TGT tickets, however, and so forces the client processes holding these tickets to request new ones from the KDC. This is done transparently to the user, so there is no service interruption. However, the problem with changing the krbtgt password is that now you know the password, and if you know it then somebody else might find out what it is. This could result in a security breach.

Kerberos Ticket Details

There are two types of Kerberos tickets: a TGT and a standard ticket. The two tickets have the same structure. The only difference is the way that they’re used. A TGT is issued during an Authentication service exchange. A standard ticket is issued during a Ticket-Granting service exchange.

Both Kerberos ticket types consist of a clear-text portion and an encrypted portion. The clear-text portion includes the following:

- **Ticket version number.** Windows 2000 uses Kerberos version 5.
- **Validating server name.** This is the name of the server that the user wants to access. In Windows 2000, this is the NetBIOS computer name, which doubles as the TCP/IP host name.
- **Validating server< realm.** This is the name of the Kerberos realm—Windows 2000 domain—that contains the validating server. This field is necessary because the security principal may be in a different domain from the server he is attempting to access.

The encrypted portion of the ticket includes the following:

- **Flags.** A series of flags assigned by the KDC determines how the ticket can be used. This includes permission to forward the ticket to another realm, which permits users in one domain to access servers in a trusted domain. A KDC can also put a flag on a ticket that authorizes a server to use it as a proxy for the original client when accessing another server. In Windows 2000, this is called delegation of trust.
- **Session key.** A random number generated by the KDC when it builds a ticket. The session key is the shared secret that the KDC, the client, and the validating server have in common. Keep your eye on the session key when tracing Kerberos transactions.
- **Client’s name and realm.** A Kerberos ticket forms only one part of the initial message sent to
a validating server. Another portion of the message includes the user’s name and domain (realm) in clear text. The validating server matches the encrypted information in the ticket against the clear-text information to make sure that they match. This prevents hijacking a ticket issued to one user and presenting it on behalf of another user.

- **Transited realms.** If a user needs to access a server in a different realm (domain), the local KDC cannot grant a ticket. Instead, it gets a ticket on behalf of the user from the KDC in the server’s realm. If the KDC does not know the name of the server’s realm, it refers the ticket request to a KDC in a neighboring realm in hopes that the neighboring KDC knows the right realm. There may be several realms between the user and the validating server. The *transited realms* portion of the Kerberos ticket stores the names of these intervening realms so that the validating server can verify that they are all trusted. Without this check, a seemingly valid ticket might actually be a Manchurian Candidate submitted by an untrusted realm.

- **Time stamp.** There are two entries in this field: the date and time that the ticket was issued and the date and time that it expires. Kerberos tickets are supposed to be impossible to impersonate, but they are programmed to die automatically just in case a bright bad guy figures out how to do the ticket. The expiration time is determined by a Time-To-Live (TTL) value configured at the KDC. For Windows 2000, the default TTL is eight hours. For the time stamp to work correctly, all Kerberos clients in a domain must have their times roughly in sync. Windows 2000 uses W32TIME service to synchronize time between domain controllers. Windows 2000 Kerberos allows a good bit of slop in time stamping. The recommended deviance is about two minutes. Windows 2000 allows five minutes.

- **Authorization data.** Windows 2000 uses this field in two ways: In a TGT ticket, it includes an encrypted random number called a *nonce* that acts as a backup validation between the domain controller and its client. If a domain controller cannot decrypt the nonce, it knows immediately that the user’s credentials are invalid. In a regular Kerberos ticket, Windows 2000 uses this field to store the user’s SID and the SIDs of any groups to which the user belongs. This SID information is used by the validating server to construct an access token for the user.

## Analysis of Kerberos Transactions

Windows 2000 uses Kerberos authentication in the following two situations:

- **Initial logon.** A Windows 2000 computer or a Windows 9x computer with the Active Directory add-on uses Kerberos to verify that a user has authorization to access the local computer.

- **Domain server access.** A Windows 2000 server that is a member of a Windows 2000 domain uses Kerberos to verify that a user has authorization to access the domain.

It is important to understand the details of these two authentication transactions so that you can troubleshoot password problems, trace the source of access denials, and determine whether a computer trust or domain trust has failed.

Let’s look at initial logon authentication first.
Logon Authentication

The following example traces the Kerberos transaction that occurs when a user logs on to a domain from the console of a Windows 2000 computer that is a member of the domain. Refer to Figure 6.2. Here are the key points to watch for:

- The user must present domain credentials to get access to a domain member computer.
- The user’s plain-text and encrypted password is never transmitted over the wire.
- The user can log on to any domain that is transitively trusted by the client computer’s domain. The user must have an account in that domain.
- The authentication results in obtaining a TGT, which can be submitted at a later time to get tickets to specific servers.
- The Kerberos transaction must use IP because DNS is required to locate a domain controller.
- The end result of a successful logon is obtaining local access to the Windows 2000 computer and a TGT that can be used to access servers in the domain.

Figure 6.2
Domain configuration for example logon transaction.

Domain Member Trust

When a computer is a domain member, it also uses Kerberos to authenticate with a domain controller via the NETLOGON service. The domain controller uses the computer’s account in the Active Directory to validate the computer’s identity. This account is virtually identical to a user account, including the SID and any group memberships associated with the computer.

Because the computer is an authorized security principal in the domain, any communication links it makes to its domain controller can be trusted by the user. One example of this is the secure RPC connection used to carry MSRPC messages between the computer and its domain controller. In the example, a successful computer domain logon is assumed. The Kerberos authentication for the initial computer logon works the same as the initial user authentication traced in the example.

Procedure 6.2 Kerberos User Logon Transaction

1. After the local computer has completed its domain authentication, the WINLOGON service presents a Welcome window.

2. The user initiates a logon sequence by pressing Ctrl+Alt+Del. This is meant to foil any Trojan
horse programs that may be impersonating the operating system at this point. Both Windows 2000 and classic NT trap the Ctrl+Alt+Del interrupt and use it to initiate a security sequence. In the case of an initial logon, the result is to have WINLOGON display a logon window.

3. The user enters an account name and password and selects a domain in the Domain field of the logon window. The Domain field can contain the following types of entries:

   o **Domain name(s).** The list includes the name of the domain to which this computer belongs and any domains trusted by that domain or other domains trusted transitively by those domains. The user must have an account in the chosen domain. The user is not required to specify a logon context—that is, the name of the organizational unit (OU) containing the user account. The names of all security principals in a domain must be unique.

   o **Local computer name.** This option permits the user to log on to the local computer without authenticating in the domain. To do this, the user must have an account in the local SAM database on the computer. Local logons still use Kerberos, but they do not involve calls to the KDC service on a domain controller. The local logon option is not available on domain controllers. The user must log on to the domain to access the console of a domain controller.

   o **Internet name.** This enables the user to enter his or her universal service name, which has this format: *username@company.com*. This shortcut provides a consistent access nomenclature. You can tell your users, "Log on using the same name as your email address."

4. WINLOGON takes the user's credentials and passes them over to the LSA SubSystem, LSASS, which works with the Kerberos security provider to encrypt the user's password into an MD4 hash function. The clear-text password disappears as soon as it is encrypted.

5. Because the user entered a domain name in the WINLOGON window, LSASS passes control over to NETLOGON. NETLOGON works with the Kerberos security provider to construct a request for a TGT. The TGT request contains the

   o User's logon ID name.

   o KDC name (Windows 2000 domain controller name).

   o Random number called a *nonce* which is encrypted using the hash of the user's password.

The nonce serves a couple of purposes. First, it enables the KDC service at the domain controller to quickly determine whether the user entered a proper password. If the KDC service cannot decrypt the nonce using the user's hashed password stored in the Active Directory, it immediately sends back an error. The nonce also acts as a laundry mark for mutual authentication. The client uses it to validate that the message is truly a response to its original message and not a bogus response generated by an impersonator.

The TGT request from the client includes a request for a particular *ticket type*. There are many ticket types, but a Kerberos client most often requests a *proxiable* TGT. Later on, if the client
redeems the TGT for a ticket to a server in trusted domain, the local domain controller can use the proxiable TGT to obtain a ticket from a domain controller in the trusted domain.

6. After NETLOGON has the TGT request, it uses DNS to determine the name of a domain controller in the target domain. This query takes the form of an SRV record request for services on TCP port 88, the well-known Kerberos port. When NETLOGON gets the name and IP address of a domain controller, it sends the TGT request using Kerberos protocols.

Let’s stop here for a second and review the status so far. At this point, the user is still waiting at the WINLOGON window of the client computer for something to happen. The NETLOGON service has asked LSASS to produce an authentication request, and LSASS has complied with a Kerberos TGT request. NETLOGON has located a domain controller in the user’s domain and sent the TGT request to the domain controller. It is now waiting for a response.

7. The NETLOGON service at the domain controller receives the TGT request and passes it to the LSASS. The LSASS uses the Kerberos Key Distribution service, KDCSVC, to look up the user’s name in the Active Directory. If the name exists, and if the hash of the client’s password successfully decrypts the nonce provided by the client, the user’s SID and the SIDs of any groups to which the user belongs are retrieved from the Active Directory and used to build a TGT.

The TGT includes a time stamp and a TTL interval along with a session key, a random number that uniquely identifies this particular ticket. The KDC also includes a nonce encrypted with the password hash of the krbtgt account. The client must include this nonce when it uses the TGT to request a specific ticket. If the TGT that comes back from the client does not have the nonce, or if the nonce is scrambled, the KDC knows that someone is trying to impersonate the user.

The KDC service encrypts the coded portion of the TGT using the hash of the user’s password obtained from the Active Directory. This hash matches the hash calculated by LSASS at the client computer. Note that neither the user’s password nor the hash function is ever transmitted on the wire.

8. The LSASS now takes the TGT and passes it to the NETLOGON service, which sends it as a reply to the NETLOGON service at the client.

9. The NETLOGON service at the client passes the TGT to the local Kerberos security provider, which decodes the encrypted portion using the hash of the user’s password. If it cannot decode the ticket—that is, the decrypted message does not match a CRC included in the encrypted portion—Kerberos discards the TGT on the assumption that it came from a bogus domain controller or was damaged in transit.

If Kerberos can decode the ticket, it checks the copy of the client nonce to makes sure it matches the nonce it put in the TGT request. If the nonce does not match, the Kerberos provider assumes that a bogus domain controller has returned a TGT and it is discarded.

If the TGT appears valid, Kerberos extracts the session key from the ticket and puts it in a cache along with the TGT. All subsequent ticket requests to the domain controller must be accompanied by this TGT. When it expires, the client must request a new one.
10. Kerberos turns the Authorization Data portion of the message containing the user’s personal and group SIDs over to the LSASS, which uses the Security Reference Monitor in the Windows 2000 Executive to build an access token for the user. At this point, the console logon is complete. The system launches the Explorer shell, attaches the access user’s token to this process thread, and the workday begins. Any subsequent processes spawned by the user are run in the user’s security context and get the user’s access token.

**Network Resource Access Authentication**

At this point, the user has access to the local machine. Now take a look at what happens when the user tries to access a resource on another server. The key points to watch for are as follows:

- The local client must go back to its domain controller to get a specific Kerberos ticket for the target server.
- The Kerberos ticket is included in the standard SMB messages between the client and server.
- The client presents the Kerberos server ticket in all transactions, making it nearly impossible for an impersonator to step in and take over the session.

The following steps describe the process for Kerberos authentication of a user attempting to access a remote server.

**Procedure 6.3 Kerberos Authenticating User Accessing Remote Server**

1. The user opens My Network Places, navigates to the name of a server, and double-clicks the server icon.
2. The TCP/IP driver converts the server’s NetBIOS name to an IP address via a call to DNS or WINS. See Chapter 4, "Understanding NetBIOS Name Resolution" for more details.
3. After the client has an IP address for the target server, the network redirector builds an SMB message to create a session and connect to the server. It can’t send this SMB yet, however, because it does not have a Kerberos ticket for the server. It calls on LSASS to provide this information.
4. LSASS calls on the Kerberos security provider to use its TGT to build a ticket request. The ticket request specifies the name of the target server, the name of the user, an encrypted random number for a nonce, and the provider’s cached copy of the TGT. The Kerberos provider encrypts its portion of the ticket request using the session key it got from the TGT as an encryption key.
5. LSASS sends the Kerberos ticket request to the KDC service on its domain controller.
6. The KDC service at the domain controller quickly validates the TGT by checking the nonce and then double-checks the user’s identity. It then builds a ticket for the target server. The ticket includes the name of the server, time stamp and TTL information, and a session key for this ticket. The KDC service encrypts part of the ticket using the password hash for the target
server. It then bundles the server ticket into a ticket reply. The reply contains a copy of the session key for the ticket encrypted with the user’s password hash along with the nonce supplied by the user.

7. The Kerberos service at the client computer obtains the ticket reply and decodes its portion of the reply to check the nonce and extract the session key for the ticket.

It uses this information to build an access request. The access request contains the server ticket (which contains a copy of the session key encrypted with the server’s password hash) and a copy of the session key encrypted with the session key itself. This encrypted session key as an authenticator. It is designed to stymie impersonators who might try intercepting server tickets and using them to gain unauthorized access. The impersonator would need to build its own access request that included the server ticket, but it would not know the session key so it could not include a valid authenticator.

8. LSASS hands the access request over to the Windows client redirector, which includes the request in the initial SMB that it sends to the target server.

9. The target server (also called the validating server in Kerberos) extracts the access request and passes it to its own LSASS, which calls on the local Kerberos provider to validate the ticket.

10. The local Kerberos provider decodes the portion of the ticket encrypted with the server’s password hash. If the decoded message does not match the CRC in the message, it is discarded. From the decoded message, the provider extracts the session key for the ticket. It uses the session key to decode the authenticator. If the session keys match, the provider checks the time stamp to verify that the ticket is still valid.

11. If all this checks out, the target server grants access to the user. The client caches its copy of the ticket and its associated session key. It uses the cached ticket for all Kerberos transactions with the target server until the ticket expires. At that point, the client gets a new ticket from the KDC. This ongoing re-authentication is one of the major strengths of Kerberos.

**Kerberos Implementation Details**

For all the great features that Kerberos enables, very few operational requirements are involved with using it. Windows 2000 computers always use Kerberos when authenticating to other Windows 2000 computers and to a Windows 2000 domain. No actions are required on the part of an administrator to configure or manage these transactions.

Transitive trusts are enabled via Kerberos without any additional configuration required. In a Native-mode Windows 2000 domain, groups and users from domains that are trusted by the client’s domain are made available on pick lists for access control lists and group memberships lists automatically. Windows 2000 servers and clients in Mixed-mode domains use a combination of Kerberos and NTLM Challenge-Response, depending on the computer they are accessing. There is no indication in the user interface about the authentication mechanism used by a particular connection.

The KDC service and the Kerberos client service do not have their own executables and are not shown on process lists. They are loaded as a part of LSASS. The Registry parameters that control the KDC service are located in HKLM | System | CurrentControlSet | Services | KDC, but this
key contains no special control values.

The Windows 2000 Resource Kit contains two utilities, KSETUP and KTPASS, for use in configuring a domain controller to act as an MIT Kerberos 5 key distribution center. As of the time of this writing, no additional tools are included for managing Kerberos.

It’s hard to talk about Kerberos without sounding like a Madison Avenue flak. Kerberos is quick, quiet, and won’t upset your stomach. Unfortunately, the same cannot be said for the security policies that take effect after Kerberos has done its duties. The following section discusses those policies.

Configuring Security Policies

Microsoft defines a policy as a "set of rules that determine the interaction between a subject and an object." The rules associated with security policies are part of an overall policy configuration mechanism called group policies. These group policies are the next generation of policies in Windows, successors to the system policies introduced in Windows 95 and NT4. System policies consisted of a set of Registry entries controlled by a single management vehicle, the System Policy Editor, or Poledit. For example, a classic NT policy combines various Registry keys and values that change the Explorer shell into a single desktop policy.

These classic system policies have serious limitations, especially when it comes to distributing security configurations.

- System policy updates permanently change the local Registries. This is called tattooing. These Registry changes can cause serious problems. If you accidentally distribute a security update that locks users out of their computers, for example, you might end up doing 10,000 desktop visits to correct your mistake.

- System policies can only be distributed in one file, the NTCONFIG.POL file. This hampers the ability to create security policies tailored to specific user communities. System policies can be targeted at specific groups, but this creates a large NTCONFIG.POL file that is cumbersome to maintain.

- The range of Registry entries that can be changed using system policies is limited because of backward compatibility with classic NT.

System policies are still supported by Windows 2000, but have been superceded by group policies. Any system parameter that relies on a Registry entry can be controlled by group policies. These changes are made in a volatile manner and do not tattoo the local Registries. If the group policy is removed, the original Registry entries take effect.

Group policies can be used to control a wide variety of configurations, including security settings, system services, user interface parameters, and application settings. In addition, group policies can be used to control the following:

- Disk quota policies
- Encrypted file system recovery policies
Registry-based group policies that affect computer settings are applied to entries in the HKEY_Local_Machine hive. Policies that affect user settings are applied to HKEY_Current_User. These policies are applied each time the computer or user logs on, and are refreshed every 90 minutes by default until the user logs off. Some policies can be set to apply only at logoff, as well.

Group policies are delivered to computers in one of two ways:

- **Local policies.** These policies are stored on the local drive in \WINNT\System32 \GroupPolicy. They affect the local computer and users who log on to that computer.

- **Active Directory policies.** These policies are stored in two places. The majority of the policies are located in the \WINNT\Sysvol\Sysvol \<domain_name> directory on every domain controller. The second Sysvol directory is shared with the name SYSVOL. The remainder of the policies are stored in the directory in a set of Group Policy Containers (GPCs). These policies can be associated with the Domain container, a Sites container, or any OU container. See Chapter 7, "Understanding Active Directory Services," for details on these containers.

The \WINNT\Sysvol\Sysvol \<domain_name> directory is replicated to all domain controllers by the File Replication service (FRS). See Chapter 13, "Managing File Systems" for details on FRS operation. In brief, the FRS is a general-purpose data synchronization service designed to replicate file updates to targeted servers. Only updated files are copied, and a database of file operations is kept to prevent accidental erasures and overwrites. FRS requires the use of NTFS5 on all replicated volumes.

The initial release of Windows 2000 includes eight group policy types. Each policy type is configured by a server-side extension in the Group Policy snap-in. When a client receives a policy configured by a particular snap-in extension, it processes the contents using a complementary client-side DLL. Table 6.2 contains the policy snap-in extensions and their client-side DLLs.

**Table 6.2 Group Policy Extensions and Client-Side DLLs**

<table>
<thead>
<tr>
<th>Server-Side Extension</th>
<th>Implementing DLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry Administrative Templates</td>
<td>USERENV.DLL (Computers and Users)</td>
</tr>
<tr>
<td>Folder Redirection Editor</td>
<td>FDEPLOY.DLL</td>
</tr>
<tr>
<td>Scripts (Computers and Users)</td>
<td>GPTTEXT.DLL</td>
</tr>
</tbody>
</table>
### Registry Tip: Client-side extension list

The client-side extension list can be found at `HKLM \ Software \ Microsoft \ Windows NT \ CurrentVersion \ Winlogon \ GPExtensions`.

The server-side extension for security policy handling, Scesrv.dll, is part of the Services suite. The client-side extension, Scecli.dll, is part of the LSASS.

This chapter covers managing security settings group policies. It includes a functional overview of the operation of the Security Settings snap-in extension and how to configure the templates used by that extension to tailor the security policies delivered to clients in your system. Group policies affecting other user configuration parameters are covered in Chapter 16, "Managing the User Operating Environment." Software distribution, other than automated deployment of Windows 2000, lies beyond the scope of this book. Automated deployment policies are covered in Chapter 2, "Performing Upgrades and Automated Installations."

Although details about the Active Directory are not covered until chapters 7 to 10, you need to understand one AD concept before looking at the way group policies are distributed in a domain. That is the concept of **containers**.

The Active Directory is an object-oriented database. A file system is an example of such a database. Object-oriented databases are structured into hierarchies in which some objects can contain other objects. In the case of a file system made up of directory objects and file objects, a "directory" is an object that can contain other objects, whereas a "file" is an object that cannot. Objects that can contain other objects are often called container objects.

---

**Active Directory Is an LDAP Directory Service**

If you are a NetWare administrator, you are accustomed to thinking of Directory Service objects in terms of them being either container objects or leaf objects. The Active Directory is an LDAP Directory Service and does not make this distinction. Any object class in LDAP can be a container.

<table>
<thead>
<tr>
<th>Security Settings</th>
<th>SCECLI.DLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Installation</td>
<td>APPMGMTS.DLL (Computers and Users)</td>
</tr>
<tr>
<td>Disk Quota</td>
<td>DSKQUOTA.DLL</td>
</tr>
<tr>
<td>Encrypting File System</td>
<td>SCECLI.DLL</td>
</tr>
<tr>
<td>Recovery</td>
<td>IEDKCS32.DLL</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>IEDKCS32.DLL</td>
</tr>
<tr>
<td>Branding</td>
<td>GPTEXT.DLL</td>
</tr>
</tbody>
</table>

---
Each object in the directory represents a real-world item, such as a user named Gwashington, a computer named PHX-DC-01, a group named Phx_Sales, and so forth. Each of these objects is derived from a particular object class. A class defines a collection of attributes (sometimes called properties). When an object is created based on object class, it is said to be an instance of the class. If an object class included the attributes of being a nearsighted system administrator with flat feet and a paralyzing coffee addiction, the author would be an instance of that class.

The directory is structured so that objects from a given object class can contain only objects from specific classes. The basis for this determination generally derives from what makes most sense in terms of database operations. For example, it makes sense that an object of the class Subnet-Container can contain objects of the class Subnet. It does not make sense for it to contain objects of the class Computer. These structure rules determine what an Active Directory tree can look like.

The directory has four significant container classes: the Domain-DNS class, the Site class, the Organizational Unit (OU) class, and the Container class. There can only be one instance of the Domain-DNS class in a domain, and that object represents the top of the domain. There can be multiple instances of the Site class, but the use of these objects is restricted to defining directory replication boundaries. The generic Container class can be created by the system only and cannot be linked to group policies. This leaves only the OU class for use as a general-purpose container for structuring a directory.

The OU, Domain, and Site containers in a directory can be linked to group policies. When a policy is linked to a container, all computer and user objects under that container inherit the policy. It is possible to modify the same Registry setting in policies linked to different containers. Local group policies can also change settings, as can legacy system policies from Ntconfig.pol. The following policy hierarchy is used to set precedence for conflicting policy entries:

- OU policies. These are applied last and have the highest precedence.
- Domain policies
- Site policies
- Local policies
- System policies

If you are a domain user who sets your Local policy to have a green desktop background but there is a Site policy that sets a blue background and a Domain policy for a yellow background and an OU policy for purple, you will get a purple desktop.

Every Active Directory has two default group policies (unless they have been deleted after installation): the Default Domain Policy linked to the Domain container, and the Default Domain Controller policy linked to the domain controller OU. You can modify these policies to set your own security entries or create new policies of your own. You can have as many policies as you want, constrained only by your willingness to manage them all.
If multiple policies are linked to the same container, they are normally applied in the order that they were created. The order can be changed in the management console by moving a particular policy in relation to the others on the list.

The files used to support group policies are not actually stored in the directory database. They are stored as discrete policy folders under `\WINNT\Sysvol\Sysvol\<domain_name>`. This is not necessarily the same `\WINNT` as `%systemroot%`. The location of the Sysvol directory is selected when a server is promoted to a domain controller.

The second Sysvol in the path is shared as SYSVOL. Client computers and users access the SYSVOL share on their authenticating domain controller to find policies. The directory contains special objects that have pointers to these policy folders in SYSVOL. These objects are instances of the class `GroupPolicyContainer`, so they are sometimes called GPC objects. GPC objects have attributes that contain the UNC path of its associated policy, the snap-in extensions used to create it, and the client-side extensions necessary to handle it.

When Windows 2000 clients log on to the domain, they query the directory for GPC objects. If any of the GPC objects are applicable to the client, the associated policy folder contents are downloaded from SYSVOL. If the policy affects Registry entries, and most of them do in one way or another, the policy entries overlay the existing Registry entries. If the policy is removed or changed, the underlying Registry entries take effect again.

Here are the key points to remember about directory-based group policies:

- The directory holds pointers to group policies stored in SYSVOL on each domain controller.
- Group policies are downloaded automatically by Windows 2000 domain members.
- Policies are applied using a hierarchy in which the OU user or computer’s OU has the highest precedent.

Before looking at specific details for group policies that affect security settings, take a look at how to view and configure those policies using the Group Policy Editor.

**Group Policy Editor Overview**

Policies are configured using the Group Policy Editor snap-in, GPEDIT.MSC. This snap-in relies on the Group Policy Editor support file, GPEDIT.DLL. This snap-in is part of several different consoles, depending on the location of the policy to be edited.

- GPEDIT.MSC is loaded into its own console to edit local policies.
- AD Users and Computers, DSA.MSC, is used to create and edit profiles associated with the Domain container and any OU containers.
- AD Sites and Services, DSSITE.MSC, is used to create and edit profiles associated with Site containers.
The Group Policy Editor has several snap-in extensions corresponding to the policy types that can be edited. All extensions are loaded by default. This procedure covers how to selectively load snap-in extensions to tailor a console to a specific purpose.

**Procedure 6.4 Editing Local Policies Using the Group Policy Editor Console**

1. Open the Run command using **Start | Run**.

2. Enter GPEDIT.MSC and click **OK**. The Group Policy console opens as shown in **Figure 6.3**.

**Figure 6.3**

Group Policy console showing Password policies.

3. Expand the tree to find the entries under the Computer Configuration icon and the User Configuration icon. These entries contain the security settings as well as the other policy paraphernalia, such as logon/logoff scripts and administrative templates for controlling the Registry configurations.

The Registry updates associated with *Computer Configuration* entries are applied when the computer starts and affect settings in `HKEY_Local_Machine`. The Registry updates associated with *User Configuration* entries are applied when a user logs on to the computer and affect settings in `HKEY_Current_User`.

Computer Configuration entries associated with Registry entries are applied when the computer starts and normally affect settings in `HKEY_Local_Machine`. User Configuration entries are applied when a user logs on to the computer and normally affect settings in `HKEY_Current_User`.

To edit just the Security Settings for a local computer, load the Security Policy console from the start menu using **START | PROGRAMS | ADMINISTRATIVE TOOLS | LOCAL SECURITY POLICY**. The Security Policy window opens. The Security Settings are the same as those loaded by GPEDIT.MSC.

To edit domain group policies, you must first know which container you want to associate with the policy. The following example uses the AD Users and Computers console to open the default domain group policy associated with the Domain-DNS container. You can use the same steps to open or create group policies for an OU container. You can also use the AD Sites and Services console to access group policies associated with Site containers. The policy options are the same. Only their place in the hierarchy differs.

**Procedure 6.5 Editing Domain Policies on Domain Controllers**

1. Open the AD Users and Computers console or the AD Sites and Services console using **START | PROGRAMS | ADMINISTRATIVE TOOLS**.

2. Right-click the Domain container at the top of the tree and select PROPERTIES from the fly-out menu. The Properties window opens. If you wanted to open or create a policy for an OU, right-click the OU icon. You cannot create policies for any other object class in this console.

4. Check the list of options under each of the Settings objects. Along with the same options that were listed for a standalone computer, there are additional entries for application deployment and central control of profile information. Set the focus on the Security Settings under Computer Configuration.

If you only want to configure one or two options, you can build a custom Group Policy console and load only those snap-in extensions you want to manage. You can use custom consoles to help farm out your administrative tasks. You may want to give security policy administration to one group, for example, and application distribution policy administration to another group. You can create a custom console for each set of tasks. Chapter 10, "Managing Active Directory Security," covers how to delegate object rights to limit administrative privileges to conform to the custom console you build. The following example describes how to build an MMC console.

**Procedure 6.6 Building an MMC Console**

1. Open an empty MMC console using START | RUN | MMC.

2. From the MMC CONSOLE menu, select CONSOLE | ADD/REMOVE SNAP-IN.

3. Click Add at the Add/Remove Snap-in window. The Add Standalone Snap-in window appears.


   The default Group Policy Object is the Local Computer. This is not a GPO as such, but looks directly at the local policy database in \WINNT\System32\GroupPolicy (see Figure 6.4).

5. Click Browse to open the Browse for a Group Policy Object window. By default, all GPOs in the directory are kept in a container called Policies, but for purposes of display they are shown under their associated OU. GPOs are linked to containers via attributes in the container objects (see Figure 6.5).

6. Select the Default Domain Policy and click OK to return to the Select Group Policy Object window. The Default Domain Policy now appears under Group Policy Object.

7. Click Finish to add the policy and return to the Add Standalone Snap-In window.

8. Click Close to close the window and return to the Add/Remove Snap-in window.
9. Select the **Extensions** tab. **Figure 6.6** shows an example.

**Figure 6.6**
Add/Remove Snap-in window showing list of available extensions.

10. Uncheck the **Add All Extensions** option and check only the **Security Settings** option.

11. Click **OK** to save the change and return to the console.

12. From the MMC CONSOLE menu (right below the title bar), select **CONSOLE | OPTIONS**. The **Options** window opens.

13. Verify that **Always Open Console Files in Author Mode** is unchecked. Author mode permits a user to add and delete snap-ins from the console.

14. Select the **Console** tab. Give the icon a descriptive name, such as Domain Security Policy Editor. This does not name the associated MSC file. That comes later.

15. **In the Console Mode combo box**, select **User Mode Full Access, Single Window**. This option prevents the user from loading additional snap-ins and extensions, but gives the user the ability to change the window’s focus. **Limited Access** maintains the look and feel of the console by preventing the user from opening new windows.

   The **Do Not Save Changes To This Console** option does not preserve the user’s icon selections and tree expansions between sessions.

16. Click **OK** to save the changes and return to the console.

17. Close the console. You’ll be prompted to save it. Give the console a filename that is descriptive but short, such as DOMSECEDIT.MSC. Save the file to the All Users profile if you want it to show in the menu for anyone logging on to this computer.

18. Open the console you just saved. See that the **Console** menu is not visible. This means the console has not opened in Author mode. If you find yourself wanting to make changes to the console at a later time, you can open it in Author mode from the command line as follows: `mmc domsecedit.msc /a`. This trick works only if you have administrator privileges.

19. Close the console.

**Functional Overview of Security Policies**

Local security policy settings are stored on the local drive in the Security Editor database, SECEDIT.SDB. This database is located in `\WINNT\Security\Database`. The SECEDIT database uses Jet technology but cannot be edited with standard Jet database tools such as Microsoft Access. The Jet support files, such as transaction logs and the checkpoint file, are located in the parent directory, `\WINNT\Security`.

The SECEDIT database gets its initial configuration settings from one of several template files. These
default templates are stored in `\WINNT\INF`.

- **DEFLTSV.INF.** Default server template
- **DEFLTWK.INF.** Default workstation template
- **DEFLTDC.INF.** Default domain controller template
- **DCUP.INF.** Domain controller upgrade template used when upgrading NT4 domain controllers

In addition to the default templates, additional templates are stored under `\WINNT\Security\Templates`. These templates contain prepackaged settings for basic, secure, and high security configurations. Any of these security templates can be loaded into the SECEDIT database and used to modify the system settings.

When you view the security entries in the Group Policy Editor, each icon under Security Settings represents a discrete section of the SECEDIT database. The template files are also divided into these sections. Figure 6.7 show these icons and the settings.

**Figure 6.7**
Custom GPE Console showing major Security Setting icons.

When you make a change to a policy setting in the Group Policy Editor, the update is written to the SECEDIT database. The WINLOGON service uses these settings to update the appropriate Registry entries. Under normal circumstances, WINLOGON would not perform this function until logoff or again at logon. If you want to force WINLOGON to load a policy change, you can use the command-line Security Editor utility, SECEDIT. The syntax is as follows:

```
secedit /refreshpolicy machine_policy (user_policy)
```

The `machine_policy` option flushes the Computer Configuration settings from SECEDIT to the HKEY_Local_Machine hive. The `user_policy` option flushes the User Configuration settings to HKEY_Current_User. This policy flush can take a minute or so. Watch the hard drive light. You can also check the Event log for an entry from SCECLI that the Group Policy entry was successfully applied.

Any changes to the SECEDIT database are also written to the associated template file so that the database can be reinitialized with the same settings, if necessary.

---

**Registry Tip: Current template file**

The name of the current template file is stored in the Registry under `HKLM | Software | Microsoft | Windows NT | CurrentVersion | Secedit | TemplateUsed`.

---

In addition to the local security settings, domain member servers and workstations download group
policy updates from the domain during logon and logoff. When a Windows 2000 member computer authenticates, it queries the Active Directory for any GPC objects. These GPC objects have attributes that point to disk-based group policy folders stored on the domain controller under the SYSVOL share, `\<domain_name>\Sysvol\<domain_name>\Policies`. The policy folder names use the globally unique identifier (GUID) assigned to the policy.

Security settings policy files for are stored in the associated policy folder under `\WINNT\Sysvol\Sysvol<domain_name>\Policies\Machine\Microsoft\Windows NT\SecEdit`. The policy filename is GPTTMPL.INF. (You may also see a GPTTMPL.PNF file, which is a compiled version of the INF.)

---

### Configuring User Security Settings

A limited number of user security settings can also be configured using the Group Policy Editor. These settings are stored in `\WINNT\Sysvol\Sysvol<domain_name>\Policies\User\Microsoft\Windows NT\SecEdit` and are downloaded to the user’s personal profile under `\Documents and Settings<logonID>\My Documents\Security\Database`. Personal settings can also be configured in a SECEDIT.SDB file and put in this folder.

---

SYSVOL can hold a long list of policy folders. The folders are tracked in the local Registry under the key `HKLM | Software | Microsoft | Windows | CurrentVersion | Group Policy | Shadow`. Each policy is assigned a sequential number starting with zero. This sequence determines the order in which policies linked to a particular container are applied, so the higher number has the higher precedence. You can change this order using the MMC console associated with the group policies you are managing.

When a client computer logs on to the domain, it downloads the GPTTMPL.INF security settings from each policy that includes that computer. For example, a domain controller would download the GPTTMPL.INF file from the Default Domain policy and the Default Domain Controller policy.

The client copies the file to the local drive under `\WINNT\Security\Templates\Policy`. Each file is first copied to a TMPGPTFL.INF file, and then renamed with a sequential number that matches the policy sequence number in the local Registry. The Default Domain policy is given a .DOM extension, and all other policy files get an .INF extension.

By default, the policy files are downloaded at logon and then refreshed every 90 minutes. You can use `secedit /refreshpolicy machine_policy` to force a download, but the security settings will not take affect until the next logon.

After the security setting policy files have been downloaded, they are layered on top of the settings in SECEDIT.SDB and applied to the Registry. This is done in a volatile manner so that the local Registry is not tattooed with permanent policy changes. You can see the difference between the local settings and the downloaded settings by opening the local Group Policy Editor, GPEDIT.MSC, or by launching the Local Security Policy console from START | PROGRAMS | ADMINISTRATIVE TOOLS. If a downloaded group policy is in force, two columns appear in the right pane, one labeled Computer Setting and one labeled Effective Setting.
Registry Tip: Security hive settings

Most of the security settings updated by group policies involve keys in the hidden and locked Security hive in the Registry. If you are following along in the Group Policy Editor, you may have noticed the Kerberos Policy settings under Account Policies. These settings are also stored in the Security hive under `SECURITY\Policy`.

You can view the keys in the Security hive on a non-production system by changing the permissions on the Security key to Administrators | Full Control and flowing the changes down the tree.

Here are the key points to remember about the way security policies are applied to local systems:

- Security policies for a local machine are stored in the Security Editor database, `\WINNT\Security\Database\Secedit.sdb`.
- Group policies, including those involving security settings, are downloaded from a domain controller when a Windows 2000 computer authenticates to the domain.
- Group policies are applied in order of precedence, with the OU containing the computer account taking priority, followed by the computer’s domain affiliation then its site. Local policies come next with legacy system policies falling in place dead last.
- Group policies downloaded from a domain controller do not permanently alter local Registry settings. If the policy changes are removed, the original Registry entries take affect once again.
- The next few sections include details on the available security setting options and how to use them to control access to local computers and domains.

Configuring Access Security Policies

Some security policies have become so firmly entrenched that it would be nearly unthinkable not to use them. This includes periodic password changes, disallowing blank passwords, imposing a lockout out after a series of unsuccessful logon attempts, and so forth. It’s possible to argue the merits of these and other policies, but for the most part it is safe to say that too much security is better than too little. This section covers the various security policies available in Windows 2000 and how to implement them, starting with password and lockout policies.

Access by Non-Microsoft Computers

The encrypted authentication passwords stored in the Active Directory can be used only by Windows clients and clients configured to use Windows authentication mechanisms. Active Directory clients authenticate using Kerberos. Classic NT clients authenticate using NTLM Challenge-Response. Down-level Windows 9x and 3.1x clients use LAN Manager authentication.
The world is not comprised of purely Windows clients, as you well know. Windows 2000 supports access by UNIX clients running SAMBA, a public-domain SMB package that has ports to virtually every flavor of UNIX imaginable. Older SAMBA versions may not work with Windows 2000 because the reversibly encrypted LAN Manager password is no longer enabled, at least in a default Windows 2000 configuration. Newer SAMBA versions understand the NTLM Challenge-Response protocol and can even act as servers in a classic domain.

Macintosh users running OS versions earlier than 7.1 will also be unable to access a Windows 2000 server without reversible passwords. Microsoft provides a custom user authentication module (UAM) for Macintosh version 7.1 and above, but the package has a history of instability on earlier clients. The UAM exposes NTLM authentication only, not Kerberos. Windows 2000 supports the native Random Number Exchange UAM that has been part of AppleTalk File Protocol (AFP) since version 2.1, but you must enable reversible passwords for it to work. This also applies to Macintosh remote access via AppleTalk Remote Access Protocol (ARAP). As with SAMBA, the best course of action is to upgrade if your system can handle version 7.1 or above.

Enabling reversibly encrypted LAN Manager passwords (also called clear-text passwords) is not recommended because of their extreme vulnerability. If you have no other alternative, you can choose between enabling them for all users in a domain or for individual users to limit your security exposure. Configure the domain policy as follows:

**Procedure 6.7 Enabling Reversible Passwords for a Domain**

1. Open the AD Users and Computers console using START | PROGRAMS | ADMINISTRATIVE TOOLS.

2. Right-click the Domain icon at the top of the tree in the left pane and select PROPERTIES from the fly-out menu. The Properties window opens.

3. Select the Group Policy tab.

4. Double-click the Default Domain Policy listing to open the Group Policy console.

5. Expand the tree to Computer Configuration | Security Settings | Account Policies | Password Policy.


7. Click OK to save the change and return to the Group Policy console.

8. Clients can now log on using their clear-text passwords. Users with pre-existing Windows 2000 accounts must log off, then log on again to create the reversible password.

If you want to enable reversibly encrypted passwords for selected users, proceed as follows:

**Procedure 6.8 Enabling Reversible Passwords for Selected Users**
1. For domain users, open the AD Users and Computers console using START | PROGRAMS | ADMINISTRATIVE TOOLS. Expand the tree to show the Users container or an OU you have configured to hold user objects.

2. For user accounts on a standalone server or workstation, open the Computer Management Console using START | PROGRAMS | ADMINISTRATIVE TOOLS. Expand the tree to System Tools | Local Users and Groups | Users.

3. Double-click the user object to open the Properties window.

4. Select the Account tab.

5. Under Account options, check Save password as encrypted clear text. Figure 6.8 shows an example.

6. Click OK to save the change and return to the console. The user must log off, and then log on to create the reversible password.

Figure 6.8
User Properties window showing clear-text option enabled.

Password Synchronization

Nothing seems to irritate users more than passwords. Users are human—not always patient, kind, and calm humans, but humans nonetheless—and whether out of forgetfulness or sheer perversity, humans do not like populating their memory with random scraps of information such as passwords. Anything administrators can do to keep users out of multiple password purgatory saves them consternation and saves administrators hours of support time.

Under normal circumstances, Windows 2000 users need only deal with one operating system password, that for their domain account. A user who has authenticated in the domain is given the Authenticated User SID, S-1-5-11, as part of his access token. By default, the Authenticated User account is a member of the Local Users group. This gives the user access to the local computer. Windows 2000 Professional also adds the Authenticated User account to the Power Users local group, a departure from classic NT4. Members of the Power Users group can install local software and create printers and do a variety of other privileged tasks not permitted to mere users.

This Authenticated User alternative to getting local access works only if the computer itself is a member of the domain. Standalone workstations and servers are not trusted to access the Active Directory on a domain controller. They can authenticate users using accounts in the local SAM only. This can be a challenge for laptop users who take their machines off the network and need to log on locally. Rather than give the user multiple accounts, one for the local computer and one for the domain, Windows 2000 caches the domain account credentials in the local SAM. This permits the user to log on locally and retain the same system privileges as if the authentication had occurred at a domain controller. This has the added benefit of retaining the user's local environment settings. Otherwise, the user would have two sets of local configurations, one associated with the domain SID and one with the local SID.
This begs the question of what to do if the computer cannot contact a domain controller and a field administrator needs to log on locally to fix the problem. Many organizations give their Windows 2000 desktops the same password for the local Administrator account. This is a questionable practice, not because it gives too much access to administrators who know the password—they could log on locally if the computer were on the network—but because the password never changes. A disgruntled ex-employee or contractor could come back a year or two after being fired and find desktops to access, either locally or across the network using the default C$ or ADMIN$ share points.

One alternative is to use random passwords for the local Administrator account and forego local troubleshooting if the computer cannot communicate with the network. Most field technicians take a dim view of this alternative. It can turn a simple network driver glitch into a time-consuming reinstallation.

Another alternative is to put a scheduled process in place at the workstations that changes the local Administrator password regularly using a file distributed to the local desktops in the background. This could be as simple as running `NET USER <username> <password>` in the context of the current local Administrator account or as complex as coding a special applet in Perl or Windows Script Host that does the job. You might want to take a look at *Windows NT Shell Scripting*, by Tim Hill (published by Macmillan Technical Publishing, 1998), for great ways to do intricate tricks like this.

Users can change their passwords on a Windows 2000 computer in the following three ways:

1. **Wait for the domain password to expire and change it when prompted.** Domain passwords expire every 42 days by default. The user is notified 14 days prior to password expiration to change the password. Both of these settings can be changed using group policies. To change the expiration interval, open the Group Policy Editor and look for `Computer Configuration | Windows Settings | Security Settings | Account Policies | Password Policy | Maximum Password Age`. To change the notification interval, look for `Computer Configuration | Windows Settings | Security Settings | Local Policies | Security Options | Prompt User To Change Password Before Expiration`.

2. **Press Ctrl+Alt+Del to bring up the Windows Security window, and then click Change Password.** The user must enter the existing password, and then the new password with a confirmation. If the user has logged on to a remote domain using a transitive trust, the correct domain should be displayed under `Log On To`. If the computer is configured with a second network client in addition to the Windows Networking client, such as Client Services for NetWare, both passwords can be changed at once.

3. **Open a command session and enter `NET USER username password /domain`.** This option is available to administrators only, and it is the fastest way to change a password. No confirmation is required, so be sure you spell correctly.

Users at down-level computers such as Windows 9x and Windows 3.1 can log on to a domain, but their computers do not have a domain, so the local system must authenticate them locally. Password changes initiated using the Password applet in Control Panel will keep the local and domain passwords in sync so long as the user keeps the default options in the `Change Password` window. If the passwords get out of sync, the user is forced to enter two passwords in the morning. If the user forgets one, he will not get access. Local Windows passwords are encrypted and then stored in a PWL file in the Windows directory. If the user's local password gets out of sync with the domain
password, the simplest fix is to delete the PWL file, log off, and then have the user log on. The system prompts for a Windows password and saves it in a new PWL file.

If the user presses the pesky Esc key to bypass domain logon, the Password Change option in Control Panel is disabled. This is good news because it prevents the user from changing only one of the passwords. Your troubles aren’t over, however. Now it’s time to learn how to configure the domain to require complex passwords. Instead of choosing a password such as Schlamiel3, the user might enter the name of a pet dog, Sam, instead. The domain rejects this password, but Windows 9x is much more forgiving and changes the local Windows password. The next time the user logs on, he becomes as disoriented as a visitor from Sirius. He pounds at the keyboard in mounting frustration, eventually picking up an intruder lockout. He then gets on the phone to the Help Desk to complain about his *^% password and your ^*& system.

Complex Passwords

On the X-Files, a hacker can guess a DoD password in two or three tries. In a production network, it’s just about that easy to guess a user’s password. Users don’t generally assign tough passwords when left to their own discretion.

In response to several successful hacks of the SAM database, Microsoft provided a strong password filter in NT4 SP3 to enforce a complex password policy. This policy and the filter are included in Windows 2000. Complicated passwords are tough to remember and often find their way onto yellow sticky notes under mouse pads, if not right there on the side of the monitor, but they do help prevent outsiders from staging successful dictionary attacks across the network.

________________________________________

NT Crack Engines

The most popular NT crack engine attacks passwords in groups of seven characters. It works on the groups simultaneously; so if a password is 11 characters long and the last four characters happen to be letters, they will be revealed almost instantly. Then they can often be used to guess the first seven characters. Similarly, putting a non-alphanumeric character at the end of an eight-character password leaves the first seven characters open for an easy crack if they are simple characters. Once they are guessed, it doesn’t take long to figure out the complex character. Keep this in mind when defining your password strategy. The best passwords are 7 or 14 characters long.

________________________________________

Complex passwords must be at least six characters long, cannot contain the username or any part of the user’s full name as entered in the directory or SAM, and must contain characters with any three of the following attributes:

- Uppercase letters (A, B, C, and so on)
- Lowercase letters (a, b, c, and so on)
- Arabic numerals (0, 1, 2, and so on)
• Special characters and punctuation symbols (#!&^+)

These example passwords would make it through the complex filter:

• #RedDog

• Jabber8wocky@%

• Spinoza#ROCKS! (Unless the user’s name is Spinoza, in which case you might have someone with a lot to say about complex names and password duality.)

There are no facilities, either in the UI or the Registry, for changing the filter settings, even if you want to make them tighter. In TechNet article Q161990, which describes the strong password filter, Microsoft states, "If you wish to raise or lower these requirements, you must write your own .DLL."

Enable complex passwords as follows:

**Procedure 6.9 Enabling Complex Passwords**

1. Open the *Group Policy* console for the Active Directory container or local computer that you want to configure. Use the steps outlined in "Group Policy Editor Overview."


3. Double-click *Passwords Must Meet Complexity Requirements of the Installed Password Filter.* The *Policy Setting* window for this option appears.

4. Select the *Enabled* radio button.

5. Click *OK* to save the change and return to the *Group Policy* console.

6. The next time the users change their passwords, any non-complex passwords will be rejected. Be sure to brief them before implementing the policy.

**Lockout Policies**

If you do not like the idea of the bad guys hammering away at your network trying to guess user passwords, you’ll want to institute a lockout policy that disables a user’s account after repeated invalid logon attempts. A lockout policy has three elements:

• **Lockout count.** The number of invalid logon attempts permitted before the account is locked out.

• **Lockout reset time.** The interval between invalid attempts after which the lockout count is returned to zero. For example, a lockout reset time of 10 minutes would cause the lockout counter to continue incrementing for invalid logon attempts made every nine minutes.
- **Lockout duration.** The interval following a lockout after which the lockout is automatically cleared and the user can try to log on again.

Lockout policies create administrative burdens because users have a hard time remembering their passwords. If you set the lockout count too low, you or your poor colleagues at the Help Desk will get lots of phone calls. If you set the value too high, you reduce the effectiveness of the policy. By the same token, if you set the lockout reset time too short, the bad guy can set his daemon to try multiple accounts with sufficient wait between attempts to foil your lockout strategy. It’s up to you to come up with a combination of settings that gives you adequate security without imposing too much of a hardship on your user support staff.

A domain account lockout policy is enforced by the LSASS on a domain controller. It applies both to console logon attempts and network logon attempts at any member computer. A console logon is the initial domain authentication performed when a user enters credentials at the console of a member computer. Console logons in Windows 2000 are handled by the Winlogon service. A network logon occurs when a user attempts to connect to a shared resource on a server using network protocols. Windows 2000 uses the Netlogon service to handle network logons. Both Winlogon and Netlogon are part of the LSASS.

LSASS also imposes the lockout policy on other services that accept network connections. This includes FTP, Telnet, and HTTP. Repeated invalid logon attempts using any combination of these services will trip the lockout.

The lockout policy has very few exceptions. The Built-In Administrator account cannot be locked out, but this exception does not apply to other accounts with administrator privileges. Computer accounts cannot be locked out, neither can domain trust accounts.

Before you put a lockout policy in place, here’s a little something to consider. Under normal circumstances, when a user enters incorrect credentials during a console logon attempt, the system rejects the attempt without giving a clue whether it was the name or the password that was incorrect. If you put a lockout policy in place, an unauthorized user who picks up a lockout knows that he at least got the logon name right. Most organizations have logon names that follow a recognizable pattern, so this revealing clue doesn’t really make much difference, but it’s worth knowing.

You can set a lockout policy for an entire domain or particular workstations. Configure the policy as follows:

**Procedure 6.10 Lockout Policy for a Domain**

1. Open the Group Policy console for the Active Directory container or local computer that you want to configure. Use the steps outlined in "Group Policy Editor Overview."

2. Navigate to **Computer Settings | Security Settings | Account Policies | Account Lockout Policy.**

3. Double-click **Account Lockout Counter** and set the value for the number of attempts you’re willing to give a user before locking out the account. Five is a good number.
4. Click OK to save the setting. The system automatically sets default values for Lockout Account Duration and Reset Account Lockout Count After to 30 minutes apiece. You can leave these settings at their defaults or choose values that make sense for your system. If you try to set the lockout duration shorter than the reset duration, the system will prompt you to at least make them equal values.

5. Close the Group Policy console.

6. From a command prompt, run secedit /refreshpolicy machine_policy to apply the changes to the Active Directory or local SAM.

7. Verify that a lockout occurs by attempting invalid logons using any account but the Administrator account.

Auditing

As sanitation engineers and quantum physicists have long known, just because you can’t see something doesn’t mean it isn’t really there. You must assume that bad guys are constantly trying to get on your network. Even if you aren’t connected to the Internet, you have to be careful of insiders, even if your company is one big happy family where everyone goes to baseball games and Nine Inch Nails concerts together.

Any activity involving Windows 2000 security objects can be audited. Logging a lot of activity puts a load on the affected server, so choose your audit points carefully. Audit reports are written to the Security log, which can be viewed by the Programs | Administrative Tools | Event Viewer. The associated Event log, SECEVENT.EVT, is located in the WINNT\System32\Config directory. You must have Administrator privileges to open this log.

The default Event log size is 512KB, which might be too small for the Security log if you audit many events. The default behavior of the log is to overwrite older entries when the log fills up, which might obscure old sins. You can change these and other log file settings as follows:

Procedure 6.11 Log File Settings

1. Open the Event Viewer console using START | PROGRAMS | ADMINISTRATIVE TOOLS | EVENT VIEWER. You only need one option.

2. Right-click the Security Log object and select Properties from the fly-out menu. The Properties window opens.

3. Change the entry for Maximum Log Size to a value large enough to accommodate several days’ logging.

4. Under Event Log Wrapping, set the logging behavior when the log file gets full. (I usually select Do Not Overwrite Events.) Don’t forget to save the log and clear it manually every week or so. You can set a system policy that causes the server to refuse connections if the Security log gets full.
5. Click OK to save any changes and close the Properties window.

You can make the Event log configurations part of an Event Log Settings policy and push them out to member servers and workstations. This gives you a standard set of log parameters when you collect your audit information.

You can read the Security logs across the network, but you must do this one at a time unless you use a third-party tool to collect Event log entries. One of the better tools for this kind of work is Event Admin from Midwest Commerce, Inc. It puts the Event log entries into an ODBC-compliant database for viewing and reporting.

Enable auditing and set audit policies as follows:

**Procedure 6.12 Audit Policies**

1. Open the Group Policy console for the Active Directory container or local computer that you want to configure. Use the steps outlined in "Group Policy Editor Overview."

2. Select Computer Configuration | Windows Settings | Security Settings | Local Policies | Audit Policy. The following events can be audited:

   o **Account Logon.** This monitors network access to a computer via network logon and logoff. Use this policy when you want a record of accounts that access the server and what privileges they have been assigned.

   o **Account Management.** This monitors an administrator who adds, deletes, or modifies the attributes of any security principal such as users, groups, or computers. This is especially useful if you have a large and diverse IT group and you’re still getting used to the way Active Directory delegation works. You might want to make sure that the new administrator in the Chemistry lab isn’t adding the entire freshman class to the SENIOR_FACULTY group, which has access to the grade spreadsheets.

   o **Directory Services Access.** This monitors administrative access to the Active Directory.

   o **Logon Events.** This monitors console logon/logoff. This is different from Account Logon, which monitors network access.

   o **Object Access.** This monitors access to security objects such as files, directories, Registry keys, and directory objects. In most circumstances, you must also configure the individual object classes for auditing. NTFS files, for example, must be enabled for auditing via the Properties window, opened by right-clicking the file in Explorer.

   o **Policy Change.** This monitors changes in any of the policies you’ve set. With so many policies flying around in Windows 2000, I leave this audit point on all the time because it shows any user and machine policy updates that come from the domain.

   o **Privilege Use.** This monitors privileged access to resources by the system or accounts that have system privileges. For example, only administrators can open the Security log.
When you open the Security log, an entry is made in the log under Privilege Use that shows your account name and what you did that exercised a system privilege.

- **Process Tracking.** This monitors access to executable code such as exe, dll, and ocx files. This is handy for figuring out who is accessing a particular file. It can also help track down viruses, although most virus scanners offer better tools.

- **System Events.** This monitors the various system updates that occur during operation. If you’re troubleshooting a pesky service that refuses to work for some inexplicable reason, this is a great trace to follow. Used in conjunction with Process Tracking, you can follow the service to see whether it performs an illegal or disallowed activity or asks the system to perform such an activity. The trace also shows you how the various security providers get initialized.

3. Double-click the policy you want to enable. This opens the Security Policy Setting window for the policy. For example, you may want to enable Audit Account Logon Events and Audit Logon Events to see the kind of reports that auditing provides.

4. Select Define These Policy Settings, and then select Success and/or Failure under Audit These Attempts.

5. Click OK to save the change and return to the Group Policy console.

6. Close the console.

7. Refresh the policy by opening a command session and entering `secedit /refreshpolicy machine_policy`. The audit policy takes effect immediately. There is no need to restart.

8. Test the audit policies you set by performing an auditable activity. For example, with Audit Account Logon Events and Audit Logon Events enabled, log on at a member server or workstation. **Figure 6.9** shows the Event Viewer console with a set of logon/logoff events under Security Log.

**Figure 6.9**
Event Viewer showing Security Log with various auditing events displayed.

9. Double-click one of the entries to see the information supplied by the auditing policy. **Figure 6.10** shows an example of the audit report for a user console logon. Notice that the user was granted an Authentication ticket that was validated by the krbtgt account.

**Figure 6.10**
Event log entry for a user console logon.

10. Close all windows and consoles.

If you don’t see any entries in the Event log after enabling auditing and running SECEDIT, press F5 or select refresh from the console menu. If you still do not get entries, and the server is a member of a Windows 2000 domain, the Default Domain Controllers group policy might have a No Auditing option set. The OU policy takes precedence over the domain policy you just set. Open the Group
Policy console for the Domain Controller OU and verify that there are no Audit policies in place. If this is not the problem, make sure that Block Policy Inheritance is not selected at the Domain Controllers Properties window, Group Policy tab. This would prevent the Default Domain policy from being inherited by the domain controller.

Assigning System Rights

Many Windows 2000 operations come accompanied by the caveat, "You need Administrator rights to do this" or "This requires Backup Operator permissions" and so forth. System-level rights are assigned using security groups that define privileges ranging from the ability to create a paging file to permission for logging on at the console of a server. Users and groups get these system rights by being associated with the applicable security group. If you are an experienced NT administrator, you're accustomed to configuring rights with User Manager. In Windows 2000, these rights are configured with the Group Policy Editor.

Before examining the list of rights in detail, take a look at how to configure the policies that control access to them.

Procedure 6.13 Configuring User Rights Policies

1. Open the AD Users and Computers console and navigate to Windows Settings | Security Settings | Local Policies | User Rights Assignment. The available policies are listed in the right pane of the window. Figure 6.11 shows an example.

2. Double-click a policy to open its Security Policy Setting window.

3. Click Add. The Group Name window opens.

   Click Browse. The Select Users or Groups window opens.

4. Double-click on groups and users from the list to add them to the name list in the bottom pane of the window.

5. Click OK to save your selections and return to the Group Name window. The names appear as a semicolon-delimited list under Audit Policy.

   Click OK to close the window. The group list is put in the main pane of the Security Policy Setting window.

6. Click OK to save this configuration and return to the Group Policy console.

Now take a look at how to apply the User Rights policies. Some of these policies deal with obscure memory and process handling that lie beyond the scope of this book. Here is a quick list of the commonly configured rights followed by details for each one along with the default groups assigned to the policy. The name starting with Se in parentheses next to each policy is the formal privilege name. You will see these names in command-line interfaces and Event log entries:
• Access this computer from the network
• Act as part of the operating system
• Add workstations to domain
• Backup files and directories
• Create a pagefile
• Bypass traverse checking
• Change system time
• Log on locally
• Shut down the system
• Force shutdown from a remote system
• Load and unload device drivers
• Manage auditing and Security log
• Take ownership of files and other objects
• Log on as a batch job and log on as a service

**Access this computer from the network.** *(SeNetworkLogonRight)* (Administrators, Everyone, Power Users, IUSR, IWAM) This right gives users network access to resources such as shared folders, printers, and Web services. The Internet User (IUSR) and Internet Web Application Manager (IWAM) accounts are added to this list when IIS is installed. Some administrators with Windows 2000 workstations that are configured as peer servers (the default) often use this privilege to restrict peer file sharing. For example, you could delete the Everyone and Power Users groups and create a group called PEER_ADMINS to put on list for this right. By controlling the membership of PEER_ADMINS, you control who has the right to share files locally at their workstations. You can also use this right to restrict file-and-print access to NT servers that are supposed to be used only for application servers. User files tend to show up in the darndest places, but to find them tucked away on your big Oracle server is aggravating. Remove the Everyone and Power Users group from this right to prevent that from happening. Users can still access the client/server application via the client.

**Act as part of the operating system.** *(SetcbPrivilege)* (Empty by default) Ordinarily you would never give this privilege to a regular user account. It is designed primarily for services that run in the background (daemons in UNIX jargon, NLMs for you NetWare folks). Vendors of these service packages, which are usually either utilities or client/server applications, design their installation routines to create a special user that is then used to control the background processes kicked off by the package. The installation routine should add that special account to the list of users with the Act As Part Of ... privilege automatically, but some are not quite that sophisticated. This is especially
true of home-brew applications. Designers in your organization might be whiz-bangs with Win32 code, but not so savvy about NT system administration.

**Add workstations to domain.** *(SeMachineAccountPrivilege) (Empty by default)* As discussed Chapter 1, "Installing and Configuring Windows 2000," a workstation must be joined to a domain before users log on to that computer with their domain passwords. This is true of both classic NT and Windows 2000. New computers come on to a network regularly, and old computers often get renamed, so it becomes a logistical problem keeping track of computer registrations. You might want to give the right to register computers to technicians at the loading bay or to department power users who receive the computers. For example, you could create a group called DEPT_GURUS and put that group on the access list for Add Workstations To Domain. You could then put selected users in this group and they would then inherit the right to register workstations into the domain without having any other system or domain privileges. This right is only effective when enabled at domain controllers; if you maintain a separate GPO for the Domain Controllers OU, you should configure this policy for that GPO.

**Back up files and directories / restore files and directories.** *(SeBackupPrivilege, SeRestorePrivilege) (Administrators, Backup Operators)* The privilege of performing backups and restores is not necessarily an honor, but it is a key security assignment. The person who can copy confidential files from a server onto backup tape, and put those files back on some other server, essentially becomes the most trusted person in that organization. It’s surprising to find organizations that require rigorous background checks for people who touch money at the front desk but perform only the most cursory review of the people doing their backups. At any rate, all third-party backup applications for Windows 2000 require you to create a special account that is put on the access list for these two rights and controls access to the files while the backup runs. If the backup fails, often it is because someone changed the password for this account or changed the name or deleted it because he didn’t know what CHEY_AGENT was used for.

**Create a pagefile.** *(SeCreatePagefilePrivilege) (Administrators)* Windows 2000 Setup creates a paging file that should be sufficient for most users. One problem that often arises, however, is fragmentation of the paging file. This has a tremendous impact on performance, much more than fragmentation of the other files on the system. The defragger that comes with Windows 2000, a pared down version of Executive Software’s Diskeeper, does not defrag the paging file. The commercial version of Diskeeper does defrag the paging file, but that costs money. One workaround is to configure a security option (covered later) that automatically deletes the paging file whenever the user logs off, and then create it anew at logon. This is also a good way to preserve security if you are concerned about one user pawing through another user’s paging file with a sector reader. If you implement this policy, you will need to add the Everyone group to the member list for this privilege.

**Bypass traverse checking.** *(SeChangeNotifyPrivilege) (Everyone)* It is not uncommon to have a highly restricted directory nestled partway down a directory tree. Many servers have restricted folders that run like limestone layers across many different directory trees in a volume. This usually happens in a panic after an audit. By bypassing traverse checking, users can skip over these restricted folders and get to unrestricted folders further down the tree. POSIX compatibility demands that the Everyone group be removed from the access list for this right. *(C2 configuration requirements does not require any change to the default configuration. You must disable POSIX for C2.)* Don’t take the Everyone group off the access list for this right until you are thoroughly familiar with the directory structure on the server and have ferreted out all pockets of highly restricted folders and moved them to discrete locations.
Change system time. (SeSystemtimePrivilege) (Administrators, [Server Operators] [Power Users]) Quite a few network processes and procedures depend on a single coordinated time source, so you might ask yourself why Microsoft didn't incorporate an automatic time sync into a domain client, a la NetWare. This remains one of the great secrets of the ages. You must run a logon script with a NET TIME command to sync the local workstation time with a domain controller. The syntax is net time /domain:dom_name /set /yes. The /yes bypasses the confirmation prompt. You must make your users a member of the Power Users group on their local workstations for this to work. Rights are always local. You can configure a domain policy to give the Users security group SeSystemTimePrivilege.

Log on locally. (SeInteractiveLogonRight) (Administrators, Account Operators, Backup Operators, Print Operators, Server Operators, IUSR, IWAM, LDAP_Anonymous, [Power Users], and [Users]) This right says, in effect, "You have permission to log on at the console of this computer." Obviously, the list for this privilege doesn't exactly define an exclusive club. The only default use of this privilege is to keep Users and Power Users from logging on at the console of server. If they try, they get a Insufficient Privileges error. This is also a common error for administrators new to trust relationships. Just because you can select your home domain from the pick list in WINLOGON doesn't necessarily mean you have local logon permissions in a trusting domain. Administrators in the trusting domain must give you SeInteractiveLogonRight permission by adding your account or a group containing your account to the Log On Locally list in their domain. The special accounts IUSER, LDAP_Anonymous, and IWAM are given SeInteractiveLogonRight because of an interesting quirk in NETLOGON. The NETLOGON service works cheek-by-jowl with the network redirectors to pass along authentication requests to either MSV1_0 or Kerberos, depending on the client. But a user who browses a Windows 2000–hosted Web site or does an LDAP lookup for an ILS connection doesn't have an account in the domain or on the host server. He or she is given access via one of the special accounts. Because these accounts do not access the server via NETLOGON, they cannot be authenticated via the network. They come in the front door as if they did a console logon. Be very careful when granting other system privileges to these accounts. You might find yourself opening a security hole without realizing it. The IUSR account is also a member of the Guests group, for example; so if you or your colleagues have a habit of enabling the Guest account and giving the Guests group access to certain directories, you might end up inviting the great Internet bourgeoisie on to your network.

Shut down the system. (SeShutdownPrivilege) (Administrators, Backup Operators, [Power Users], [Users]) Under most circumstances, you do not want an average user to walk up the console of a server, press Ctrl+Alt+Del and select Shutdown. For this reason, only Administrators and Backup Operators have this permission for servers.

Force shutdown from a remote system. (SeRemoteShutdownPrivilege) (Administrators, Server Operators, [Power Users]) It's often necessary to reboot a system. Windows 2000 is much less obnoxious in this regard, but you'll still find yourself rebooting a server to clear out memory or free up a seized application. And you might need to restart a user's system if the user isn't in the area to give a manual three-finger salute. You can use the Shutdown utility that comes in the NT Resource Kit (there's a graphic version, SHUTGUI, for those of you who don't like command-line utilities), but you first need to have the Force Shutdown right at the target computer.

Load and unload device drivers. (SeLoadDriverPrivilege) (Administrators) If there were ever a
right that you do not want to distribute to the average user population, this is it. Enough said.

**Manage auditing and Security log.** *(SeSecurityPrivilege) (Administrators)* This right gives permission to view, save, and clear the Security log and to establish audit policies. If you want to give your corporate or campus security officers the right to manage auditing on your Windows 2000 servers without giving them wide-ranging powers, you can put them on this access list.

**Take ownership of files and other objects.** *(SeTakeOwnershipPrivilege) (Administrators)* Every security object in NT has an owner. This is a fundamental tenet of C2 that is implemented right at the gut level of Windows 2000. An owner can do anything with the object he or she owns. Period. End of story. As an Administrator with *SeTakeOwnershipPrivilege*, you can take ownership of an object away from another user. It is also possible to assign ownership to another user, but not with the standard tools in Windows 2000. One possibility is to use a utility called *chown* from Mortise Kerns Systems, [http://www.mks.com/](http://www.mks.com/). A demo version of this and other UNIX-like tools comes in the Services for UNIX toolkit from Microsoft. Using *chown*, you can also assign ownership groups as well as individual user. For example, you could establish an ORPHAN_DATA group to assign ownership of files and directories that have been abandoned by their original users. Such a group would be easy to search for, thereby simplifying data cleanup.

**Log on as a batch job and Log on as a service.** *(SeBatchLogonRight) (SeServiceLogonRight) (Empty)* These two rights are somewhat complementary. It sometimes comes to pass that applications or utilities need to launch automatically and run in the background. No process can run as an orphan, and most background services run under the auspices of SYSTEM. This has its limitations, though. You don’t necessarily want SYSTEM to run a third-party application. What if the application crashes and renders SYSTEM incapable of tending to other duties? Most apps establish a user account responsible for launching and monitoring the background process. This account must get authenticated, and it does so as either a batch job or a service. The difference is one of approach. An executable must either be specially designed to be a service or parlayed into a service using a utility such as SRVANY from the NT Resource Kit. A batch job can be any BAT or CMD file, so it is much easier to set up, but you still need a host to launch it in the background. The Scheduler is just such a host. Open it from the Control Panel. You can specify a user ID to use when running the process, and Scheduler will take care of getting the correct permissions.

**Assigning Security Options**

The final major group of security policies exposed to the Group Policy Editor are security options. This list consists of individual Registry entries that have some passing applicability to security. Unlike the preceding policies, these entries are not associated with users. They affect general system operation. Here is a list of a few of the more commonly encountered policies. An in-depth explanation of their default security options, associated Registry keys, values, and functions follows the list.

- Allow system to be shut down without having to log on
- Audit the access of global system objects
- Audit the use of all user rights including backup and restore
- Rename Administrator account [Guest Account]
- Clear virtual memory pagefile when system shuts down
- Shut down system immediately if unable to log security audits
- Do not allow enumeration of account names and shares by anonymous users
- Disable Ctrl+Alt+Del requirement for logon
- Do not display last username in logon screen
- Secure channel: digitally sign secure channel data
- Secure channel: digitally encrypt all secure channel data
- Secure channel: digitally encrypt or sign secure channel data
- Encrypt files in the offline folders cache
- Automatically log off users when logon time expires
- Message text for users attempting to log on
- Message title for users attempting to log on
- Number Of previous logons to cache
- Restrict CD-ROM access to locally logged-on user only
- Restrict floppy access to locally logged-on user only
- Send unencrypted password to connect to third-party smb servers

**Allow system to be shut down without having to log on.** (HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ ShutdownWithoutLogon) There is a Shutdown button in the WINLOGON window that permits a user to shut down the computer without logging on. This discourages users from just shutting off the machine, although it has been my experience that few people get this message. The ShutdownWithoutLogon feature is disabled on servers so that the system can tell you whether users are accessing the server or deliver other pronouncements before you initiate the shutdown. If you find this annoying, enable this option.

**Audit the access of global system objects.** (HKLM | System | CurrentControlSet | Control | Lsa | AuditBaseObject) The interior of Windows 2000 is like the Pentagon. There are many places you don’t really want to see and most of them are pretty dull, even if you could see them. Ordinarily, if you enable object access auditing using an audit policy, these obscure internal objects will be excluded from the list. Enabling this option adds them to the list. For the most part, this is only useful to driver and system-level programmers.

**Audit the use of all user rights including backup and restore.** (HKLM | System | AuditRight)
CurrentControlSet | Control | Lsa | FullPrivilegeAuditing) This is one of those "why bother" features. Under normal circumstances, if you enable auditing of privileged object access, the Security log would fill as soon as you performed your first backup and the exercise of the Backup Operator privilege is logged over and over again. This auditing is ordinarily bypassed. When you set this option, the exercise of Backup and Restore will be logged. Only use this when you think that someone is misusing his or her privileges. And be sure to make the Security log very large to accommodate the entries and be prepared for a very slow backup.

**Rename Administrator account [Guest Account].** (Registry or Directory entry locked) You have to assume that the bogeyman lurks everywhere, and you can be sure that the bogeyman is after your Administrator password. You cannot delete Built-In accounts and groups—they are as fixed as the Statue of Liberty—but you can change their names, and many administrators do to make it that much more difficult for the bogeyman to try dictionary attacks. This security option can make the change for you and propagate it throughout your enterprise. If you decide to implement this policy, make absolutely sure you have several other accounts with full administrative privileges, and then change the names to something long and incomprehensible. Then, write down the new names and passwords and keep them safe. You may need them. Not the Guest account, certainly, but NT treats the Administrator account, or rather the SID that represents that account, with special respect. He's like a J.R.R. Tolkien wizard. There are some privileges hidden from view that only the one true Administrator can wield.

**Clear virtual memory pagefile when system shuts down.** (HKLM | System | CurrentControlSet | Control | Session Manager | Memory Management | ClearPageFileAtShutdown) As discussed earlier in this chapter, clearing out PAGEFILE.SYS is a good way not only to enforce security but to prevent performance-robbing fragmentation of the paging file. It requires some extra time to clear out the file, but not much. This option is highly recommended, especially if you do not deploy a third-party defragger that works on the paging file.

**Shut down system immediately if unable to log security audits.** (HKLM | System | CurrentControlSet | Control | Lsa | CrashOnAuditFail) If you choose to audit a system, you might object to someone logging on if that audit stream is not tracking. If the Security log fills, the bad guy might sneak on unnoticed. The ultimate stop to this problem is to shut down the system as soon as the Security log runs out of room. This may strike you as somewhat self-destructive way to enforce security audits, but hey, you have to grant the effectiveness. If you choose to enable this option and the system crashes, you can restart and logon as Administrator—not an account with Administrator privileges, mind you, only the Administrator account will work. If you renamed the Administrator account, this is one situation where you'll need the special account.

**Do not allow enumeration of account names and shares by anonymous users.** (HKLM | System | CurrentControlSet | Control | Lsa | RestrictAnonymous) Consider this scenario. You've created a classic NT one-way trust relationship between a resource domain and an account domain. Standard stuff for a master domain model. You're an administrator in the resource domain and you want to put a global group from the master domain on to the access control list on a local directory. You have logged on to the local resource domain with your administrator password. You make contact with the master domain using the NTFS permissions tool to get a list of users and groups. At that point, you are not logged on with an account from the account domain. How do you get that user list? Well, the domain controller gives domain special permission to anonymous members of the resource domain to enumerate the user list. You can disable this permission using this option if you don't want to give the resource domain administrators a list of users. This is most commonly done
when the trust is established to a client’s domain.

**Disable Ctrl+Alt+Del requirement for logon.** *(HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | DisableCAD)* This new addition to Windows 2000 was intended to give workstation users who are not members of a domain the ability get right to the Explorer shell without a logon. It should not be enabled for member workstations or servers.

**Do not display last username in logon screen.** *(HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | DontDisplayLastUserName)* It’s handy not to be forced to type in your username each time you log on, but some users (and some security-conscious administrators) object to leaving the name exposed so that the bad guy can guess at a password.

**Secure channel: digitally sign secure channel.** *(HKLM | System | CurrentControlSet | Services | Netlogon | Parameters | SealSecureChannel)* This and the following two settings are designed to deal with a very specific potential security breach in Windows 2000. Member workstations and servers communicate to their domain controllers over a secure RPC link. This link is not checked for integrity, so a bright bad guy could conceivably impersonate a machine on the wire and divert secure traffic. The digital signing identifies each packet coming over the secure link. It slows down communications somewhat. (I do not have statistics.)

**Secure channel: digitally encrypt all secure channel data.** *(HKLM | System | CurrentControlSet | Services | Netlogon | Parameters | SignSecureChannel)* Same as the preceding description, but with the addition of encryption for all traffic, not just passwords.

**Secure channel: digitally encrypt or sign secure channel.** *(HKLM | System | CurrentControlSet | Services | Netlogon | Parameters | RequireSignOrSeal)* Same as preceding description, but forces secure traffic instead of letting the members auto-negotiate. Only set this option if every domain controller is set the same way.

**Encrypt files in the offline folders cache.** *(HKLM | Software | Microsoft | Windows | CurrentVersion | NetCache | EncryptEntireCache)* Client-side caching speeds network communication by holding copies of executables and other read-only files at the local machine. A bad guy could conceivably nab these saved files and steal valuable secrets. Encrypting the files slows performance, but improves security. You may have noticed a trend.

**Automatically log off users when logon time expires.** *(HKLM | System | CurrentControlSet | Control | Session Manager | ProtectionMode)* You can define a specific time of day after which a user or group of users is denied access. This is useful when you have a group of users who have specific duties from 8 to 5, but should not be messing around on the network after hours. Many managers find this option attractive in the era of Internet browsing. The question becomes, what happens to the user who is on the network when the fatal hour approaches. If you do not set this option, the user is warned repeatedly that their time has expired but they can continue to use network resources. If you enable this option, the user is warned several times and then, at the scheduled time, all network access is cut off. If you use this option, be sure to warn your users that the warnings mean business. They had best save their work or it will be lost. The users are not forced off the local machine. This policy affects network access only.

**Message text for users attempting to log on.** *(HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | LegalNoticeText)* This and the next option define parameters for
a special window that appears after the user presses Ctrl+Alt+Del but before the WINLOGON credentials window. It is most often used to display boilerplate HR text like "You are using COMPANY equipment and must obey COMPANY policies that are available in COMPANY offices."

**Message title for users attempting to log on.** (HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | LegalNoticeCaption) This option specifies the title bar text in the window configured by LegalNoticeText.

**Number of previous logons to cache.** (HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | CachedLogonsCount) If a Windows 2000 workstation cannot contact its domain controller, a user can still be logged on via cached credentials. The cache is set for 10 by default. If you have a workstation at which many people log on, some of them may have been aged out of the cache. You can set this option to as many as 50 logons.

**Restrict CD-ROM access to locally logged-on user only.** (HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | AllocateCDRoms) C2 security requirements include a specification to exclude removable media from network access. This and the following option meet that requirement. You may be wondering about your Jaz or Zip drive or CD-R unit. The answer is, don’t do it if you C2 it.

**Restrict floppy access to locally logged-on user only.** (HKLM | Software | Microsoft | Windows NT | CurrentVersion | Winlogon | AllocateFloppies) See preceding item.

**Send unencrypted password to connect to third-party smb servers.** (HKLM | System | CurrentControlSet | Control | Lsa | LmCompatibilityLevel) Windows 3.1x and Windows 9x clients who connect to a Windows 2000 server participate in NTLM Challenge-Response but do not use MD4 hash to share encrypted information. Instead, they use an older DES encryption with a password that actually travels along the wire. You can stop this behavior by setting this option, but only if you have no down-level clients remaining on your network.

## Loading Custom Security Templates

The security settings in SECEDIT.SDB are initialized by a template file. Templates used to initially configure a system are stored in \WINNT\INF. Additional templates are stored in \WINNT\Security\Templates. You can modify these templates or build new ones and load them into SECEDIT.SDB using the Security Templates snap-in.

You can also use a template to analyze your existing security configuration to determine whether any security problems exist. This is done using The Security Configuration And Analysis snap-in.

This section covers the steps for using both of these security snap-ins to configure system security. First, take a look at how to modify an existing template or build a new one and apply it to the local security database. This should only be necessary if you have a standalone server or workstation. A domain member computer should be managed using group policies.

**Procedure 6.14 Configuring Security Templates**
1. Open an empty **MMC** console by entering **MMC** at the **Run** window.

2. From the console menu, select **CONSOLE | ADD/REMOVE SNAP-IN**. The **Add/Remove Snap-in** window opens.

3. Click **Add**. The **Add Standalone Snap-in** window opens.


5. Close the window to return to the **Add/Remove Snap-in** window. The two snap-ins are on the list. There are no extensions.

6. Click **OK** to save the additions and return to the **MMC** console.

7. Select **CONSOLE | SAVE AS** and save the console with a descriptive name such as **SECCONF.MSC**.

8. Expand the tree for both snap-ins. **Security Templates** contains a list of files from **\WINNT\Security\Templates**. **Security Configuration and Analysis** shows a set of instructions for performing an analysis. Figure 6.12 shows an example.

![Figure 6.12](image)

**Figure 6.12**

Custom MMC console showing Security Templates and Security Configuration and Analysis snap-ins.

9. Expand the tree under one of the templates and examine the Account Policies and Local Policies settings. Each template packages a set of policy options designed to simplify security configuration.

10. To modify an existing template, expand the tree under the template icon and modify the settings.

11. To build a new template, right-click the template path and select **NEW TEMPLATE** from the fly-out menu. A window opens with fields for Template Name and Description. Enter this information and click **OK** to create the template. The template is added to the template list and contains no special settings. Configure it based on your requirements.

After you have configured a template with the settings you think are appropriate for your system, apply it to the local system by importing it into the security database. Do this as follows:

**Procedure 6.15 Applying a Custom Security Template**

1. Open the local Group Policy editor, **GPEDIT.MSC**.

2. Expand the tree to the Security Settings icon.

3. Right-click the Security Settings icon and select **IMPORT POLICY** from the fly-out menu. The
Import Policy From window opens.

4. Double-click the name of your new template to load it into the security database.

5. Verify that the local settings reflect the options you selected in the Template Settings console.

6. Close the Group Policy editor.

If the computer is a domain member, the settings you apply to the local database will not override group policies downloaded from the domain. If you want to check the current security settings for a computer, including those in the downloaded group policies, you can run an analysis using a copy of the Security Configuration database, SECEDIT.SDB. This database is locked when the system is in operation, so you must make a copy of it. The copy can be any name and can remain in the same \WINNT\Security folder.

**Procedure 6.16 Performing a Security Analysis**

1. Load the custom console that contains the Security Configuration and Analysis snap-in.

2. Right-click the Security Configuration and Analysis icon and select OPEN DATABASE from the fly-out menu. The Open Database window opens.

3. Navigate to \WINNT\Security\Database and load the copy of SECEDIT.SDB. If you try to load the SECEDIT.SDB file itself, you will get an Access Denied error.

4. With the database loaded, right-click the Security Configuration and Analysis icon and select analyze computer now from the fly-out menu. The Perform Analysis window opens with a path for the analysis log. The default path goes to the Temp directory in your local user’s profile.

5. Click OK. The analysis is performed. The Analyzing System Security window charts the progress. The event should not take long to complete unless your configuration includes a complex list of file and Registry security entries.

6. When the analysis is complete, expand the tree under Security Configuration and Analysis. It shows the standard set of security settings icons and policies. When you highlight a policy, the right pane shows a comparison of the settings in the copy of the SECEDIT database you loaded and the current computer settings. Figure 6.13 shows an example. Any differences between the Database Setting entries and the Computer Setting entries are attributable to group policies that have been downloaded from the domain.

**Figure 6.13**
Analysis results after running Security Configuration Analysis.

7. Scan the contents of the analysis and decide how you want to handle any discrepancies. The best course of action for a domain member is to modify the group policies. For a standalone server, if the database contents are correct, you can right-click the Security Configuration and Analysis icon and select CONFIGURE COMPUTER now from the fly-out menu. This will overwrite any local configuration policies with the contents of the configuration database.
8. If you want to do just the opposite, to update the configuration database with an existing template, right-click the Security Configuration and Analysis icon and select Import Template. Select the template from the list and click OK to apply it.

Configuring Secondary Logon

UNIX administrators often sneer, perhaps rightly so, at the inconvenience of the admin tools in Windows 2000 and NT. If you really want to use `grep` and `awk` and those other utilities that sound like my grandfather coughing up phlegm while he watched the Ed Sullivan show, get the Services for UNIX add-on pack. Before you pop for the extra tools, however, you might find a few in the Windows 2000 shrink-wrap that will serve your needs.

One tool addresses a security issue and a nagging problem for field technicians. The security issue is that Windows 2000/NT administrators spend too much time logged on with administrator privileges. This exposes the network to all kinds of bad germs. The administrator needs a way to stay logged on with normal system privileges but to quickly get administrator rights to perform specific functions.

The nagging problem for technicians concerns desktop visits. Let’s say you sit down at a user’s desktop and you need to perform a few diagnostic routines or install a piece of software. The user doesn’t have admin rights. You don’t want to log on with your ID because of the change in profile.

In both cases, what is needed is the ability to exercise administrative privileges—to do a quick superuser logon, if you will. Windows 2000 enables this feature with the Second Logon Service, SECLOGON. The SECLOGON service permits you to kick off a process in an administrator's context. The SECLOGON service is one of the suite of services loaded and managed by the Services executable. The SECLOGON service starts automatically at logon.

Call up the SECLOGON service to launch a program by using the `runas` command. The syntax for `runas` is: `runas /user:<domain or computer name>\username app_name`.

For example, log on to a Windows 2000 workstation as a standard user without admin rights. Open a command prompt and run the Time command. Try to change the system time. You’ll be denied access. Users cannot change the system time. This requires administrator privileges. Now enter the following command (the example uses a domain name of company):

`runas /user:company\administrator cmd`

You’ll be prompted for the password for Administrator. Enter it. A command session opens in the context of Administrator. Look at the title bar to verify this. Any applications launched from this session use the administrator's access token and system privileges. Now try to change the system time again. You will succeed.

You can also use the `runas` command to open MMC snap-ins from a user’s desktop. If you want to load the Computer Management console to check the partitioning of a user's drive, for example, you can use the following command to launch the console in the Administrator's context:

`runas /user:company\administrator "mmc c:\winnt\system32\compmgmt.msc"`
After you have finished your administrative chores, be sure to close any programs and sessions you opened with `runas`. You don’t want to leave a door open for the user.

## Moving Forward

At this point in the Windows 2000 migration odyssey, the preliminaries have been completed. We’ve installed a few Windows 2000 servers and configured name resolution with WINS and DNS. We have good control over IP addressing with DHCP. And we have control of access security and security policies. Now we’re ready to start designing the architecture of the network, and to do that we need to know a lot more about the Active Directory.

© Copyright Pearson Education. All rights reserved.