

ITPro<sup>TM</sup>  
SERIES

WindowsITPro  **eBooks**

Keeping Your Business  
SAFE from Attack:

# Passwords and Permissions

By Roger Grimes

**Microsoft<sup>®</sup>**



**Microsoft®**

## Contents

<b>Chapter 2 Windows Authentication</b> .....	<b>22</b>
<b>Why Authentication?</b> .....	<b>22</b>
<b>Authentication Fundamentals</b> .....	<b>23</b>
Identification .....	23
Authentication .....	24
Password Hashing .....	25
Authorization .....	26
Windows Logon Process .....	27
<b>Types of Windows Authentication</b> .....	<b>27</b>
Windows Logon Authentication Protocols .....	28
LAN Manager Protocol .....	29
NTLM .....	30
NTLMv2 .....	31
Kerberos .....	31
Authentication Service (AS) Exchange .....	33
Ticket-Granting Service (TGS) Exchange .....	34
Client-Server (CS) Exchange .....	34
Kerberos Policy Settings .....	35
Why Not Use Only Kerberos? .....	35
Anonymous Logons .....	36
Disabling Anonymous Enumeration .....	38
Restricting Anonymous .....	38
Windows Logon Authentication Best Practice Recommendations .....	41
Remote Authentication Protocols .....	41
Extensible Authentication Protocol (EAP) .....	42
Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) .....	43
Challenge Handshake Authentication Protocol (CHAP) .....	43
Password Authentication Protocol (PAP) and	
Shiva Password Authentication Protocol (SPAP) .....	43
Unauthenticated Access Authentication .....	44
Internet Information Service (IIS) Authentication Protocols .....	44

Anonymous Authentication .....	45
Basic Authentication .....	45
Digest Authentication .....	45
Integrated Windows Authentication .....	45
.NET Passport .....	45
SSL/TLS Digital Certificates .....	46
<b>Microsoft Credential Manager .....</b>	<b>47</b>
<b>Summary .....</b>	<b>49</b>

## Chapter 2:

# Windows Authentication

Chapter Two covers Windows authentication protocols and mechanisms in detail, with special attention to Windows Logon Authentication protocols. Internet Information Service (IIS) and remote authentication protocols are covered briefly to contrast with and to round out a fuller picture of Windows authentication.

## Why Authentication?

According to conventional wisdom, the three reasons for securing computers are confidentiality, integrity, and availability—sometimes known as the CIA Triad. Of these reasons, integrity is the most critical. In general terms, integrity refers to data and means that data has not been modified during its transmission from user to user. Ensuring integrity therefore includes more than a simple focus on data; it also includes tasks relating to authentication of both data and users. When users digitally sign data, they certify that the data is in a specific form, current as of the time they sign it. When the data arrives at the receiver and the data's digital signature is verified, the verification determines that the data did not change between the time it was signed and the time it was received.



### **Note**

**We use the term *data* to mean any type of digital bytes, whether it's data, program files, or multimedia content.**

But even more important than authenticating the data is confirming the user's identity. If the users who rely on the data cannot rely upon the identify of the sender or creator of the data, the rest of the security triad isn't material. If you can't be sure who made or sent data, what does it matter if that data is protected? Why should network administrators work to ensure high rates of data availability if they can't trust the data itself? Data security becomes important only if the data can be confirmed during creation, modification, and transmission. Data becomes valuable and therefore worth protecting when its sources can be authenticated.

Viruses and worms, and other types of malware attacks, proliferate only because the malicious hackers have a high level of assured anonymity. Their chances of being identified and charged with a crime are miniscule in today's Internet environment. Malevolent hacking would stop in an instant if hackers knew they could be identified. Devious insiders would not harm data or plant logic bombs if they knew they could always be traced.

For many companies, the strategy for reaching the goal of a more secure network—a network worth protecting—is doing away with default anonymity. Microsoft and other entities are working toward universal default authentication with a variety of proposals and protocols. Since the introduction of the NTFS subsystem, Windows computers can identify and authenticate users, computers, and other security principals on the local computer or domain. To solve the most persistent problems, companies like Microsoft are introducing new authentication and identity schemes. For example,

Microsoft's Caller ID™ and Sender Policy Framework (SPF) are two recent proposed authentication standards that are designed to reduce the amount of unsolicited email that seeps into our networks.

Perhaps the grandest proposal is Microsoft's Next Generation Secure Computing Base (NGSCB; <http://www.microsoft.com/resources/ngscb/default.msp>). NGSCB will let trusted applications and data communicate with users and each other securely. Untrusted programs, like viruses and worms, will be unable to make unauthorized changes to the system and data, which will effectively kill them off in mass. The seeds of NGSCB were implemented in Windows XP Professional Service Pack 2, but the first full implementation will appear in the next version of Windows (code-named Longhorn) and require special CPU extensions.

To summarize, authentication is used to identify a user, computer, or some other security principal reliably. Successful authentication is then used to assign access controls to different objects and to audit object use. We cannot control access to resources without an authentication mechanism.

## Authentication Fundamentals

Authentication is the process by which a security principal (a user, a group, a service, or a computer) identifies itself to an authentication authority. The security principal is a unique, defined identity in the system to which it is applying for authentication—in Windows, an account. In authentication vernacular, the account involved in the client-side of the authentication event is known as the *supplicant*. Every Windows security principal has a security identifier (SID) which uniquely identifies the principal for authentication, authorization, auditing, delegation, and trusts.

Figure 2-1 shows the basic authentication process from identification through authorization.

**Figure 2-1**

*Basic authentication*



In the next sections, we look at each step in the process, consider how password hashing fits into authentication, and review the Windows logon process before examining Windows authentication options in detail.

### Identification

The process of identification requires that a security principal submit, as proof of its identity, something that only that principal is capable of submitting—in Windows, the logon account name. Then, the security principal's identity is verified, based on the presented proof. For a principal to be authenticated, the proof submitted must equate to the expected value of this proof (also known as the *authenticator*) that is stored in the authentication database.



#### **Note**

The term *authenticator* can be used to represent two different security concepts when discussing authentication. It can mean, as it does in this paper, the proof submitted to prove identity; or it can mean the authentication mechanism and database used to authenticate.

Identity can be proved by:

- Something only the security principal knows
- Something only the security principal has
- Something only the security principal is

Something only the security principal *knows* includes passwords, passphrases, or personal identification numbers (PINs). Smart cards and tokens are examples of something only a security principal *has*. Biometric data, such as fingerprints or retinal patterns that can be scanned, is something only a security principal *is*.

These proofs of identity are known as *factors*. If an authentication process requires the security principal to use two of the three types of factors, the process is known as *two-factor authentication*. If the security principal is required to use identifiers from all three factor classes, then the process is known as *multi-factor authentication*. Two-factor authentication is considered to be more reliable than one-factor authentication (all other factors equal), and multi-factor authentication is considered the most reliable authentication method.

Email phishing attacks, in which a forged email tricks a user into revealing security credentials to an unauthorized third party, is leading many entities, such as online banks, to require two-factor authentication. Although a password or PIN may be easy to steal, it is less likely that a malicious hacker will also have the legitimate user's smart card or token at the same time. High-security companies require multi-factor authentication. VeriSign (<http://www.verisign.com>), which maintains some of the Internet's top-level domain naming servers and is involved in a significant portion of all Internet commerce, requires multi-factor authentication to access its protected computers. Users wishing to access the inner computer sanctums must type in their PIN, swipe a smart card badge, and submit to a hand geometry scan.

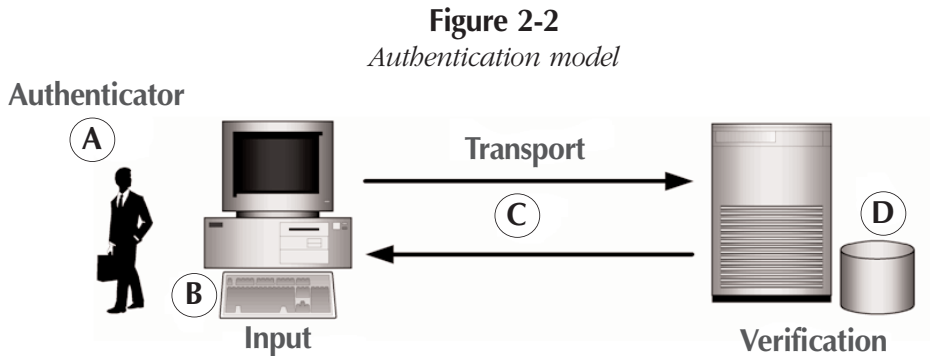
### **Authentication**

Regardless of the way the identity credentials are submitted, an authentication mechanism (that is, an authentication protocol) submits the credentials to an authentication database for comparison. In Windows, authentication databases are usually either a local security accounts management (SAM) database or an Active Directory database.

Protecting all the transactions—the transmission of the authenticator and all subsequent transactions—against unauthorized modification or eavesdropping is an important part of authentication. The goals of protecting the transactions include:

- Preventing capture of principal credentials
- Limiting exposure of principal authenticator
- Preventing manipulation of authentication traffic
- Preventing spoofing of authentication server identity
- Reducing burden on authentication server
- Providing timely information about principal authorizations

As Figure 2-2 shows, any authentication system contains an authenticator that is input and then securely transported to the authentication database for verification. The processes and mechanisms that are involved in the authentication process are called *authentication protocols*.



Authentication protocols protect the transmission of the authenticator and the resulting transactions. In early systems, when users typed their passwords, they were transmitted in clear text to the verification database. Malicious hackers can insert an eavesdropping tool, sniff the password credentials, and authenticate as the other user. Unfortunately, this breach can still happen today in unprotected FTP, Telnet, and POP authentication transactions.

Modern authentication protocols do not pass the original authenticator in clear text. Instead, the authenticator is manipulated or sent through a cryptographic process between the sender and receiver. The manipulation is done so that only the security principal, who knows the plain-text authenticator, can create the correct obscured authenticator.



### Note

For the rest of this chapter, the simpler term *password* will be used in place of *authenticator*.

## Password Hashing

Plain-text passwords are not usually stored in the verification database. As is the case on the security principal's originating computer, the password is manipulated by a cryptographic process and stored in its changed form. Thus, if the verification database is compromised, the plain-text passwords cannot be discovered immediately.

Password hashing is an example of a simple password protection mechanism. In a system that uses password hashing, the user's input password is converted, by using a hashing algorithm, into a value; this value is then stored in the verification database. Hashing algorithms are usually well-known cryptographic formulas that convert input into a unique computational value. Hashes are considered one-way in that even if you have a hash result, it is difficult to find the original value without computing all possible hash results. Some Windows authentication protocols use MD4 (<http://www.faqs.org/rfcs/rfc1320.html>) or MD5 (<http://www.faqs.org/rfcs/rfc1321.html>) hashing algorithms.

Let's look at the whole process. When users log on, they input their user account name and password. The plain-text password is computed to its password hash, and the password hash is sent to the authentication process and compared to the value stored in the verification database. If the values match, the authentication is successfully verified. The authentication protocol knows that only someone with the correct password could have created the correct password hash. With this method, passwords can still be used for authentication without the risk of transmitting them in clear text.

You might ask, "Can't the password hash be sniffed off the wire and then used to fake later authentication?" Yes. For that reason, today's authentication protocols usually don't actually send the password hash between sender and receiver; instead, they go one step further. The verification database server creates a unique one-time value, called a *challenge*, and sends it to the principal. The principal (in actuality, its authentication proxy process) uses its password hash cryptographically against the challenge to respond. The response is sent to the verification database server, which has performed the same computation. If the two resulting challenges agree, then authentication is verified. Secure authentication protocols even include a timestamp in the challenge and response to ensure that hackers can't use the resulting computations in a later replay attack. All in all, authentication protocols can be quite complex.



### Note

**To understand authentication thoroughly, it's important to understand the difference between the authentication protocol—the processes that encompass the whole authentication process—and the stored password hashes that are involved in the authentication process. Some malicious hacks attack the authentication process, but it is more common for hackers to attack the resulting password hashes in an attempt to recover the original passwords.**

The authentication process can be separated from the sender and receiver involved; this setup is called *trusted third-party authentication* (TTP). Many of today's authentication networks, like Kerberos (discussed below) and Public Key Infrastructure (PKI) rely on TTPs to store the protected password (authenticator) credentials. In this way, protected credentials don't have to be stored on every single sender and receiver that could potentially perform authentication. Also, you can increase security on the TTP to prevent unauthorized access, which is easier to do on one, or a few, centralized computers than across all computers.

## Authorization

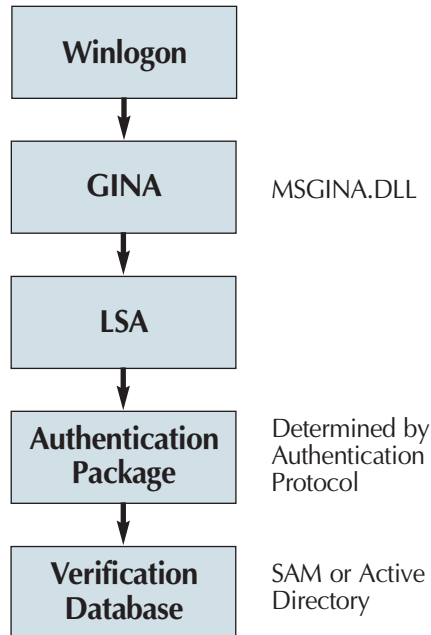
After a successful authentication, Windows gathers the cumulative permissions assigned to the principal's account and security group memberships, and the security principal is authorized to access allowed resources. However, authorization doesn't mean that the system won't perform more identifications and authentications—authentication occurs more often than only during the logon process. Nearly every time a principal accesses a Windows object, an authentication event is triggered.

For example, when a user connects to another computer's file or print share, the user must be authenticated. When the security principal—in this case, a user—contacts the share, the principal is challenged for its authentication and must pass an authentication process or be vouched for by a verification database server.

## Windows Logon Process

Now let's look at the specific logon process that Windows uses before we cover the Windows authentication protocols in detail. Figure 2-3 shows the interactive logon (in this case, a local logon) process.

**Figure 2-3**  
*Windows interactive logon process*



When a user initiates an interactive logon (a local logon with the Ctrl-Alt-Del sequence), the Winlogon service calls the Graphical Identification and Authentication (GINA) program. The GINA extracts the user's logon name, password, and domain information and passes them to the local security authority (LSA). The LSA program uses the authentication protocol as a guide to create authentication packages that are sent either to the local or to the remote verification databases.

## Types of Windows Authentication

Windows can use several different types of authentication, depending on the access you need and the location of the security principal. The most popular categories of Windows authentication are Windows Logon authentication, remote user authentication, and Internet Information Services (IIS) user authentication. Each has strengths and areas in which an administrator must use caution. Highlights for each type of authentication are summarized in Figure 2-4; we cover each in detail in the next sections.

**Figure 2-4**  
*Windows authentication protocols*

### Windows Logon Authentication

**LM**—old, insecure

**NTLM**—pre-Win2K default

**NTLMv2**—Win2k and later, plus NT SP4, Win9x with AD client (NTLM(v2) still used in NT domains even when Win2K logs into NT domain)

**Kerberos**—default Win2K protocol, can't be used by earlier clients

### Remote User Authentication

**Protected EAP (PEAP)**—adds TLS tunnel layer to other EAP types, commonly used in wireless auth.

**EAP-TLS**—smart cards or certificates.

**EAP-MD5**—uses CHAP, uses passwords.

**MS-CHAPv2**—Win2K and later, and older clients with DUN upgrade can VPN (but not dial-up), mutual authentication. **MS-CHAP**

**CHAP**—plain text, supports old and non-Windows clients. Server must allow reversible encryption.

**SPAP**—plain-text login, not tested.

**PAP**—plain-text logon, user can't change password easily.

### IIS 6.0 User Authentication

**Anonymous**—uses IUSR\_servername account

**Integrated Windows Authentication**—no second login, good for local, uses NTLM or Kerberos

**Digest Authentication**—sends hash auth., IIS must reside in DC domain, user still logs in, not as secure as IWA

**Basic**—plain text login dialog box

**.NET Passport** (Certificates-machine auth., with trusted CA)

## Windows Logon Authentication Protocols

Windows Logon authentication happens whenever a user or computer logs on to a Windows computer or domain, or whenever an object with security access control entries (ACEs) is contacted. The Windows Logon Authentication protocols are

- LAN Manager
- NTLM (version 1.0)
- NTLMv2
- Kerberos



### Note

The abbreviation **NTLM** refers to version 1.0, whereas **NTLMv2** refers uniquely to the second version of NTLM.

With increasing preference, each protocol in the list above is more secure and resistant to malicious attack than the one listed before it. LAN Manager is the weakest and Kerberos is the most secure. Most Windows computers support all four protocols, depending on the circumstances. Let's look at each protocol in detail. To make the discussion of each easier, we will look at authentication from the perspective of only an interactive logon. Other authentication events are similar.

## LAN Manager Protocol

LAN Manager (LM) protocol was created by IBM and implemented in early versions of Windows and Windows networking. Like all Microsoft authentication protocols, LM converts typed-in plain-text passwords into password hashes that are stored and used by sender and receiver during the authentication process. This process is called the LM hash.

LM password hashes are created by performing the following steps on the submitted plain-text password:

1. All alphabetic characters are converted to uppercase.
2. The password is padded or truncated to make it exactly 14 characters long, and then it is broken into 7-character halves.
3. Each half is used as a DES key to encrypt a constant string.
4. The two results are concatenated into a 128-bit string and storing it as 32-byte hex string.

The LM password hash is then used in the LM authentication protocol process in the following way:

1. The client makes the authentication request to the server.
2. The server generates a 16-byte random number as a challenge and sends it to the client (Type message).
3. On the client, the LM hash is created, as described in the paragraph above, from the plain-text password typed into the GINA GUI.
4. The 16-byte LM hash is null-padded to 21 bytes.
5. This value is split into three 7-byte thirds.
6. Each 7-byte third is used to create a DES key.
7. Each DES key is used to DES-encrypt the challenge from the Type 2 message sent by the authenticating server, resulting in three 8-byte ciphertext values.
8. These three ciphertext values are concatenated to form a 24-byte value, which is the client's LM protocol response.

Although the LM protocol was adequate protection when it was invented, it is easy to exploit using today's technology and techniques. Among its problems is the relatively small number of possible passwords created by converting all lowercase characters to uppercase and limiting passwords to 14 characters. With the computing power available today, all the possible password hashes are easy to pre-compute in a short time.

Worse, if the password isn't 14 characters long, which is almost always the case, the added spaces were easily recognizable in the hash result. Thus, a hacker could immediately spot the exact size of the password and begin hacking on just that password size, focusing only on uppercase alphabetic characters and numbers.

To exacerbate the problem, most users used only alphabet characters—no numbers, symbols, or spaces—and most spelled common words that could be found in the dictionary. The possible resulting passwords could usually be cracked by only a few thousand guesses. In the world of authentication, that kind of security isn't even close to being good enough. For more details, see <http://www.insecure.org/spl0its/10phtcrack.lanman.problems.html>.

### NTLM

Because of the weaknesses in the LM protocol, Microsoft created the NTLM protocol as the default authentication protocol for Windows NT 4.0. NTLM uses a more sophisticated authentication algorithm and creates a more secure stored password hash. NTLM can support up to 128 character passwords. Unlike LM hashing, which is very limited in the ASCII characters that can be used, NTLM hashing supports the full Unicode character set, thereby adding more complexity to the process.

The NTLM protocol uses the following steps to create the password hash.

1. The password is truncated at the 128th character.
2. The password is converted to 16-bit Unicode value.
3. The value is run through the MD4 digest function.
4. The resulting 128-bit MD4 hash is stored as 32-character hexadecimal string.

On the downside, NTLM performs absolutely no salting. *Salting* is the process of changing the outcome of a password hash by using a randomly created string of characters. Without salting, every identical password has an identical password hash. If a hacker can pre-compute a large collection of password hashes, a captured password hash might be cracked to its originating plain-text password.

Here is the NTLM authentication process between a security principal and a server:

1. The client makes authentication request to server.
2. The server generates a 16-byte random number as a challenge and sends it to the client (Type 2 message).
3. The client creates an NTLM hash of password.
4. The 16-byte NTLM hash is null-padded to 21 bytes.
5. This value is split into three 7-byte thirds.
6. These values are used to create three DES keys (one from each 7-byte third).
7. Each of these keys is used to DES-encrypt the challenge from the Type 2 message (resulting in three 8-byte ciphertext values).
8. These three ciphertext values are concatenated to form a 24-byte value. This is the NTLM response.
9. The client sends NTLM response to server.
10. The server sends three items to a verification database for authentication: the username, the challenge sent to the client, and the response received from client.
11. The verification process retrieves the hash associated with the user and uses the hash to encrypt the challenge.
12. If the result and the client's response match, authentication is successful and the verification database responds to the original server.

For more information about the NTLM protocol, check the following sites:

- <http://davenport.sourceforge.net/ntlm.html>
- [http://www.securityfriday.com/Topics/ntlm\\_optimizedattacks.html](http://www.securityfriday.com/Topics/ntlm_optimizedattacks.html)
- [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/Security/microsoft\\_ntlm.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/Security/microsoft_ntlm.asp)

## NTLMv2

Security researchers eventually proved that the NTLM protocol was insecure. Microsoft responded by creating NTLM Version 2 (NTLMv2) and released it with Windows NT 4.0 Service Pack 4. NTLMv2, although surpassed by Kerberos as today's default Windows authentication protocol, has proven to be reliably secure for most environments.

The NTLMv2 protocol is similar to NTLM, except that

1. HMAC-MD5 message authentication is used in the NTLMv2 password hash
2. A block of data called the *blob* is constructed with timestamp, client challenge, and other info and used in the process
3. The challenge and the blob are concatenated to form NTLMv2 response

Here is the NTLMv2 authentication process between a security principal and a server:

1. The client makes an authentication request to a server.
2. The server sends the client a challenge in a Type 2 message.
3. The NTLM password hash is created (as discussed previously, this is the MD4 digest of the Unicode mixed-case password).
4. The Unicode uppercase username is concatenated with the Unicode uppercase authentication target (domain or server name).
5. The HMAC-MD5 message authentication code algorithm (described in RFC 2104) is applied to this value using the 16-byte NTLM hash as the key. This results in a 16-byte value—the NTLMv2 hash.
6. The blob—a block of data—is built from a timestamp, a client challenge, and other information from the Type 2 message.
7. The challenge from the Type 2 message is concatenated with the blob. The HMAC-MD5 message authentication code algorithm is applied to this value using the 16-byte NTLMv2 hash (calculated in step 2) as the key. This results in a 16-byte output value.
8. This value is concatenated with the blob to form the NTLMv2 response.

Overall, NTLMv2 is harder to compromise with brute-force attacks than NTLM because it uses a 128-bit encryption key. Most attacks against NTLMv2 are successful only because of weak passwords (brute force attacks always work against weak passwords) and because older authentication methods (LM, NTLM) can be forced by a client (who could be an attacker).

## Kerberos

Kerberos is the default logon authentication protocol in Windows 2000 and later. It is a free mutual authentication and encryption method developed by MIT and supported by most major vendors, including Microsoft. The name is taken from Greek mythology, referring to the three-headed dog that guarded the gates of Hades.

Kerberos uses symmetric key encryption technology and is considered extremely secure because the security principal's password hash (actually the challenge response) is transmitted over the network only during the initial authentication event. After that, the TTP server authenticates security principals to requested resources with temporary session keys. The benefits of Kerberos include:

- Mutual authentication
- Increased password protection
- Replay attack protection
- Delegation of identity
- Interoperability

In the previous authentication protocols, the security principal was always authenticating itself to the verification database and/or requested resource. Kerberos requires mutual authentication, in which the resource must also prove its identity to the requesting client. The temporary session keys are encoded with timestamps, so that attackers cannot capture and replay the session keys at a later time. Kerberos also allows a principal to assign its credentials to another computer. Lastly, because Kerberos is an open, widely supported standard, Microsoft's implementation (with some adjustments) can interoperate with other vendor's implementations.

This next section will cover the Kerberos authentication process. Although it can seem a bit complicated the first few times you see it, you don't have to understand the technical mechanics behind Kerberos to use it. In fact, Kerberos works quite well on its own most of the time, and network administrators don't have to adjust it.

Let's start with a high-level summary of the Kerberos process, followed by a more detailed example.

When a client wishes accesses a Kerberos protected resource, it requires a Kerberos *ticket*. The (highly simplified) process for receiving a ticket is:

1. The The client first requests authentication from the Kerberos Authentication Server (AS)
2. The The AS authenticates the user when the user successfully logs on.
3. The The AS approves the request and creates a session key.
4. The The session key authenticates the client to the Ticket-Granting Service (TGS).
5. The The TGS gives the client a Kerberos ticket.
6. The The client hands the original resource the Kerberos ticket to authenticate itself.

Before we continue, some Kerberos terminology and technology requires definition.

- A *realm* is a Kerberos boundary for the authentication system. In the Windows world, a realm usually equates to a domain.
- The *Key Distribution Center* (KDC) server is sometimes referred to as either the *Authentication server* or the *Ticket Granting server*, although these are just different functions of the server, not distinct roles. In Windows, the domain controllers run the KDC service. An account named `krbtgt` is created on DCs for the Kerberos service. You can not turn off the KDC on a domain controller.
- *Tickets* are data packages issued by a KDC that contain information about the authenticated client, the resource they are authenticated to, the duration of that authentication, the session key, and other data. A user ticket is what is traditionally called a ticket granting ticket (TGT).
- A *session ticket* is also called a *service ticket*.
- The *long-term key* or *secret key* is derived from the user's password. Long-term keys for computers are more complex (14 characters) and are automatically changed every 7 days.
- A *session key* is generated by the KDC and is good only for the particular authentication session between principals.

- A *Privilege Attribute Certificate* (PAC) contains the SIDs associated with the user or groups the user belongs to.

The Kerberos protocol is composed of three subprotocols. The subprotocol in which the KDC server gives the client a logon session key and a TGT is called the Authentication Service (AS) Exchange. The subprotocol in which the KDC distributes a service session key and a ticket for the service is called the TGS Exchange. The subprotocol in which the client pre-sends the ticket for admission to a service is called the Client/server (CS) Exchange.

To understand the Kerberos protocol in more depth, let's look at what happens with each subprotocol. Here is the chain of communication involved in a Kerberos authentication session between a client workstation and a resource server in a Windows 2000 domain environment; we will follow a request through

- The AS Exchange
- TGS Exchange
- C/S Exchange.

This overview is still somewhat simplified, but it will give you an accurate picture.

### **Authentication Service (AS) Exchange**

The starting point: User A wants access to a Windows 2000 network.

1. User A, at a Microsoft Windows 2000 Professional workstation, logs on to a Microsoft Windows 2000 network, typing the user name and password.
2. The Kerberos client running on A's workstation converts the password to an encryption key and saves the result in a program variable.
3. The Kerberos client sends a message of type KRB\_AS\_REQ (Kerberos Authentication Server Request) to the KDC. This message has two parts:
  - An identification of the user, A, and the service for which A is requesting credentials, the TGS.
  - Pre-authentication data, intended to prove that A knows the password. This data is simply an authenticator encrypted with A's master key. The master key is generated by running A's password through an OWF.
4. Upon receipt of KRB\_AS\_REQ from A, the KDC looks up the user A in its database (the Active Directory), gets the user's master key, decrypts the pre-authentication data, and evaluates the time stamp inside. If the time stamp passes the test, the KDC can be assured that the pre-authentication data was encrypted with A's master key and is not merely a captured replay.
5. Finally, once the KDC has verified A's identity, it will create credentials that the client program on A's workstation can present to the TGS. The credentials are created and deployed as follows:
  - A brand new logon session key is encrypted with A's master key.
  - A second copy of the logon session key and A's authorization data, in a Ticket Granting Ticket (TGT), is encrypted with the KDC's own master key.
  - The KDC sends these credentials back to the client by replying with a message of type KRB\_AS\_REP (Kerberos Authentication Response)
  - When the client receives the reply, it decrypts the logon session key by applying A's master key. The session key is then stored in the client workstation's ticket cache. The TGT is extracted from the message and stored in the cache as well.

### **Ticket-Granting Service (TGS) Exchange**

Here, the starting point is a request from the Kerberos client running on User A's workstation actually requests credentials to access the target server, User B.

1. The Kerberos client running on A's workstation sends a message of type KRB\_TGS\_REQ (Kerberos Ticket-Granting Service Request) to the KDC. This message consists of the following components.
  - The identity of the target service for which the client is requesting credentials
  - The authenticator encrypted with the user's logon session key
  - The TGT acquired from the AS Exchange
2. The KDC decrypts the TGT with its master key and extracts A's logon session key. A's logon session key is used to decrypt A's authenticator. If A's authenticator passes the test, the KDC invents a new session key for A to share with B. Two copies of this new session key are sent back to A in a single message, encrypted as follows:
  - One copy is encrypted using A's logon session key.
  - The second copy is encrypted using the target server's master key, in a ticket along with A's authorization data.
3. User A's client decrypts the target server session key, using its logon session key, and stores the session key in the local cache, along with the target server ticket.

### **Client-Server (CS) Exchange**

The Kerberos client on User A's workstation is now ready to be authenticated by the target server, User B.

1. User A's Kerberos client sends User B a message of type KRB\_AP\_REQ (Kerberos Application Request). This message contains
  - An authenticator encrypted with the session key for B.
  - The ticket for sessions with B, encrypted with B's master key.
  - A flag indicating whether the client requests mutual authentication.
2. B decrypts the ticket and extracts A's authorization data and session key. B uses the session key to decrypt A's authenticator and evaluates the time stamp. If the authenticator passes the test, B looks for a mutual authentication flag. If this flag is set, B uses the session key to encrypt the time from A's authenticator and returns the result to A in a message of type KRB\_AP\_REP (Kerberos Application Reply).
3. A's client decrypts the reply with the session key. If the authenticator is identical to the one sent to B, the client is assured that the server is genuine, and the connection proceeds.

As you can see, Kerberos is a complex protocol, but it is a secure one. Microsoft's implementation of Kerberos has withstood many tests. To date, only one vulnerability has been demonstrated (<http://ntsecurity.nu/toolbox/kerbcrack>), and it has not been widely deployed. The attack is known as *Kerbcrack*; it works by capturing and cracking the less-protected pre-authentication packets which are sent, ironically, to prevent password-sniffing attacks.

### Kerberos Policy Settings

Kerberos authentication can be customized with the policy settings listed below.

- Enforce user logon restrictions: your options are Yes and No
- Maximum lifetime of service tickets: the default is 10 hours
- Maximum lifetime of user tickets: the default is 10 hours
- Maximum lifetime for user ticket renewal: the default is 7 days
- Maximum tolerance for computer clock synch: the default is 5 minutes

Checking logon restrictions determines whether the user's account has been locked out since the user ticket was originally issued. It also makes the resource server check to make sure the user has been granted permission to access the computer from the network.

Particular problems may arise that force administrators to change the default values, but those situations are not common. In general, don't change the default settings unless you have a good justification and fully understand the implications. For example, changing the maximum lifetime of service tickets from 10 hours to 5 hours could result in twice the normal amount of authentication traffic on the participating domain controllers.



#### Note

**Windows built-in smart card domain authentication requires Kerberos.**

### Why Not Use Only Kerberos?

If Kerberos is the most reliable authentication protocol, why not use only Kerberos? The short answer is this: Windows can't always use it. Although Kerberos is the default login authentication protocol for Windows 2000 and later, in several circumstances, older password hashes and protocols are still commonly used.

- LM password hashes are stored by default on all Windows systems, even Windows Server 2003. When you hear about password-cracking tools easily breaking into modern Windows computers, they are usually breaking into the much easier to hack stored LM password hashes, not the NTLM or Kerberos protocols.
- LM must be used to connect to LAN Manager networks or older legacy clients, such as OS/2, DOS, Mac, WFW, or Win95, that have not been updated to support new authentication methods.
- NTLM is supported on Windows NT, Windows 2000, and Windows 2003.
- NTLMv2 is supported natively in Windows 2000, Windows XP, and Windows Server 2003 and can be added to Windows NT and Windows 98 with the Active Directory Client Extension software. It can be added to Windows 95 and 9x clients using DFS client, WinSock 2.0 update, or Microsoft DUN 1.3 (KB 239869).
- NTLMv2 is used in mixed-mode domains.
- You can't use Kerberos with RRAS; NTLMv2 is the user authentication protocol that is used.
- When an NTLM challenge is issued by a server to a Windows client, all Windows clients, by default, reply with both an NTLM (or an NTLMv2) and an LM response.
- Windows 2000 and later computers still use NTLM or NTLMv2 for authentication when authenticating to computers outside its own domain (unless in a Windows 2003 forest).

- When a computer joins a Windows 2000 (or later version) domain for the first time, NTLM or NTLMv2 is used.
- For circumstances when domains aren't set up, such as logging in locally or peer-to-peer connections, NTLM or NTLMv2 is used.
- NTLM or NTLMv2 is used when connecting to non-Kerberos versions of Windows or to resources of servers that aren't domain members.
- Telnet and FTP services use NTLM or NTLMv2.

As you see, it is virtually impossible to rid yourself of all authentication protocols prior to Kerberos. Windows 2000 and later systems will always use a combination of Kerberos and one or more of the other authentication protocols. However, you should disable LM and NTLM protocols when possible. You can use group policy settings or registry modifications to force the Windows logon authentication protocols used by clients and servers.

The group policy setting for network security is LAN Manager Authentication Level. In the list below, the value number is associated with the value in the LMCompatibility registry key that could be modified: The possible values for the LAN Manager Authentication Level are

- 0 = Send LM and NTLM responses (the default for the Windows XP Local Security Policy). Clients use only LM and NTLM (not NTLMv2).
- 1 = Send LM and NTLM; use NTLMv2 session security if negotiated. Clients first use LM and NTLM, using NTLMv2 session security if the server supports it.
- 2 = Send NTLM response only (the default for the Windows Server 2003 Default Domain Controller Policy). Clients use NTLM and NTLMv2 session security if the server supports it.
- 3 = Send NTLMv2 response only. Clients use NTLMv2 authentication only and NTLMv2 session security if the server supports it.
- 4 = Send NTLMv2 response only \ refuse LM. Clients use NTLMv2 authentication only and NTLMv2 session security if server supports it.
- 5 = Send NTLMv2 response only \ refuse LM & NTLM. Clients use NTLMv2 authentication only and NTLMv2 session security if server supports it.

These settings affect client computers and domain controllers differently. Only the last two settings listed affect DCs by limiting them to NTLM and NTLMv2 (option 4) or just NTLMv2 (option 5). Otherwise they also accept LM, NTLM, or NTLMv2.



### **Note**

**This setting doesn't affect Kerberos use.**

### **Anonymous Logons**

No discussion of Windows authentication would be complete without mentioning anonymous logons. In many instances, Windows needs to have unauthenticated access to a Windows system to do its requested work. For this reason, Microsoft created a non-credentialed session called the *anonymous logon* or *null session*.

The anonymous user account has no password, but it can be controlled through access control lists. For example, you can give the anonymous user account the ability to access any file or folder by selecting it when you define security permissions. The anonymous user account is not part of the Authenticated Users group. The anonymous user is part of the Everyone group in Windows 2000, but not in XP or 2003.



### Note

**Anonymous logons have nothing to do with IIS's anonymous user accounts (for example, IUSR\_computername) even though they sound similar.**

If you want to create and experiment with an anonymous user session, follow these steps:

1. Get to a command prompt.
2. Use the Net View command to enumerate shares on a remote computer. For example, type **Net View \\<remotecomputersname>**. A list of the computer's advertised shares should appear. If anonymous enumerations are not allowed, the query asking to list that computer's shares will fail. If so, configure the computer (at least temporarily) to allow anonymous enumeration. You can do it in group policy and reboot or find the share and explicitly give it anonymous permissions.
3. Now establish an anonymous connection to the remote computer. For example, type **Net Use \\<remotecomputersname>\<sharename> "" /user:""** This will establish an anonymous connection.
4. Issue the same Net **View \\<remotecomputersname>** command again. This time, it should work, demonstrating the anonymous account's newly established connection.

Anonymous logons have been the subject of controversy since the early days of Windows NT 4.0 because the earlier versions of Windows allowed and relied upon anonymous enumeration. The word *enumeration* is synonymous with *query*; it is the way Windows systems discover various objects. It uses TCP/IP ports 139 and 445. For example, when Network Neighborhood is used to browse, enumeration is used locate computers, network information, and shares. Another good example occurs when accounts and resources are accessing resources in other domains. An anonymous enumeration event is necessary even for the basic trust information to be negotiated and verified—for example, creating a list of valid user accounts that should be accessing the resource domain.

Of course, unauthorized users can attempt enumeration to learn about a system to attack it. Using anonymous enumeration, hackers can gather the following information about a Windows system:

- Network Information (domain, trusts, etc.)
- Shares
- Users and groups
- Privileged accounts
- Registry keys

Hackers who learn enumeration information, such as drive shares and user names, can use that information to attack the network. For example, anonymous enumeration may let a hacker identify

user accounts as administrative or non-administrative by looking at the user account's SID. By default, user accounts are given well known SIDs that identify privileged accounts. For example, the SID S-1-5-xxx-500 is always the administrator account; SID S-1-5-xxx-519 is always the enterprise administrator account; and S-1-5-xxx-512 is always the domain administrator account. A common defense against this attack is to rename the administrator account to something that a hacker would not automatically guess. However, if the computer system allows anonymous SID enumeration, a remote hacker may be able to query the user accounts and locate the renamed administrator. If hackers can enumerate your network, half their work is done.

### **Disabling Anonymous Enumeration**

However, disabling all anonymous enumeration isn't the answer, either. It can cause problems with legacy operating systems and applications, as listed below.

- Pre-2000 users may not be able to change passwords after they expire
- Adding workstations to domains may fail
- Trust problems with NT domains may arise
- Browser service may experience problems
- Exchange GAL problems may come up.

### **Restricting Anonymous**

The more important security defense is to restrict anonymous access instead of disabling it outright. The RestrictAnonymous feature (and registry key) was added in NT4 SP3. The values and behaviors are slightly different in NT, 2000, and later operating systems. Compared to previous versions, Windows 2003 has enhanced granularity of control. Although anonymous enumeration can be restricted with the Restrict Anonymous registry key (HKLM\System\CurrentControlSet\Control\LSA\RestrictAnonymous), it is easiest to use a Group Policy setting (Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options) to accomplish the same thing. The Restrict Anonymous registry key values, and the behavior they set, are listed below.

- In Windows 2000 and XP
  - 0 = anonymous enumeration allowed
  - 1 = anonymous enumeration of SAM accounts and shares allowed only using certain Win32 API calls
  - 2 = anonymous enumeration not allowed; no access allowed without specific permissions given
- In Windows 2003
  - 0 = anonymous enumeration allowed
  - Any non-zero number = anonymous enumeration of SAM accounts and shares allowed only using certain Win32 API calls

In Windows 2000 and later, setting RestrictAnonymous=1 lets programs that use various legitimate APIs perform anonymous enumerations. Therefore, if you want to prevent all anonymous enumeration in 2000, set the value to 2 (unfortunately, setting the value to 2 doesn't work in Server 2003 or XP). Setting the value to 2 in Windows 2000 removes the Everyone SID from the anonymous user, which is handled by a separate key in XP and 2003. Because preventing ALL anonymous enumerations can disrupt legacy operating systems and applications, most people should not set

Restrict Anonymous to 2. In XP and 2003, any non-zero number (including 2) has the same effect as =1. Even with values set at 1 or 2, explicit permission for the anonymous user account can be given to different registry keys, shares, Named Pipes, etc. This setting simply prevents anonymous enumeration without explicit access.

Never set the Restrict Anonymous registry key value to 2 in mixed domain environments or with legacy clients. It is safer to set the value to 1. If you need to permit clients running versions of Windows earlier than Windows 2000 to change their passwords, add the Everyone and Anonymous Logon groups to the pre-Windows 2000 compatible access group, enabling anonymous access to the accounts. Use the *Let Everyone permissions apply to anonymous users* policy to extend anonymous access to match the Windows NT 4.0 model. Disable or do not configure this policy if your domain includes computers running versions of Windows earlier than Windows 2000 or if it has an outbound, one-way trust relationship with a domain in another forest.

The browser service on computers running Windows NT 4.0 and earlier requires the ability to enumerate shares anonymously when it connects to backup browsers, master browsers, and domain master browsers to retrieve server lists and domain lists. Users on the trusting side of a one-way trust relationship need the ability to enumerate SAM accounts anonymously when they add domain accounts and groups on the trusted side of the relationship to security groups in the trusting domain. Most, but not all, of the issues are related to pre-2000 computers and domains.

It is easier to control anonymous enumeration by adjusting group policy settings in Windows Server 2003. The settings are

- Do not allow anonymous enumeration of SAM accounts
- Do not allow anonymous enumeration of SAM accounts and shares
- Allow Anonymous SID/Name translation

The first option, *Do not allow anonymous enumeration of SAM accounts*, was called *Additional restrictions for anonymous users* in Windows 2000. *Do not allow anonymous enumeration of SAM accounts and shares* is a new option in XP and 2003. *Allow Anonymous SID/Name translation* is a new option in XP and 2003; it was not part of 2000.

*Allow Anonymous SID/Name translation* is disabled by default on Windows Server 2003 member servers (but not domain controllers). If enabled, it makes a system **less** secure because it lets hackers find renamed administrator and guest accounts. Unfortunately, domain controllers must leave anonymous SID/Name translation enabled in order to do their job.



### Note

**Because domain controllers have no SAM to worry about, disallowing anonymous enumeration of SAM accounts has no effect on domain controllers.**

One of the big changes between 2000 and later versions (XP and 2003) is that anonymous users are not part of the Everyone group, by default. You can, however, enable the *Let Everyone permissions apply to anonymous users* setting to support legacy applications and operating system functions. The anonymous user can never be added to the Authenticated Users group, which has the same membership as the Everyone group without the anonymous user and guest account.

Windows Server 2003 has new ways to fine-tune what objects anonymous users can enumerate and to define the shares, registry keys, and named pipes that are remotely accessible. New security settings define what default shares can be accessed anonymously. The default shares include COMCFG and DFS\$. COMCFG is used for SNA legacy applications and DFS\$ is used by the Windows Distributed File System feature. The COMCFG share is not usually active on most servers (unless you have SNA server installed), so you can ignore it. If DFS is enabled on your server and the DFS\$ share is enabled, do not disable it.

The *Remote accessible registry paths and subpaths* setting is another new security setting introduced in Server 2003. It replaces XP's *Remote accessible registry paths*. Either security setting defines registry keys that are accessible to anyone, including anonymous users, regardless of the key's ACL permissions. You can add registry keys to remotely accessible registry paths as desired. Turning off remote registry accesses breaks most remote management tools.

Default settings for named pipes that can be accessed anonymously are COMNAP, COMNODE, SQL\QUERY, SPOOLSS, LLSRPC, EPMAPPER, LOCATOR, TrkWks, and TrkSvr.

- The COMNAP and COMNODE named pipes are used for SNA sessions with certain clients (OS/2, Win95, Windows 3.x and LanMan 2.2c) that are trying to connect to a Microsoft SNA Server (service) running on a Windows NT 4.0 box.
- The SQL\QUERY named pipe provides null connectivity to Microsoft's SQL Server running on a Windows NT 4.0 box.
- The SPOOLSS named pipe provides null connectivity to the Spooler service (Print Server). Typically SPOOLSS can be removed from the Print Server without causing too many problems, but problems can occur with legacy clients (see Knowledge Base article 162695 SMSINST, entitled "Access Denied" Message When Connecting to a Printer").
- The LLSRPC named pipe provides null connectivity to the RPC Interface of the License Logging Service.
- The EPMAPPER named pipe provides null connectivity to the Endpoint Mapper, which lists all the applications that are using RPC. Disabling this can cause problems.
- The LOCATOR named pipe provides null connectivity to the name-service provider service.
- TrkWks (workstation) and TrkSvr (server) are related to Distributed Link Tracking and should not be removed.

As shown in Table 2-1, the Restrict Anonymous setting is the guiding control that determines whether an anonymous connection can access local SAM accounts and shares. Note that Anonymous in the Everyone Group setting does not seem to affect the outcome.

**Table 2-1 Effect of disabling anonymous enumerations on a domain controller**

Setting		Outcome
Restrict Anonymous Enabled?	Anonymous in Everyone Group	Can Anonymous enumerate local SAM shares and accounts?
No	Yes	Yes
No	No	Yes
Yes	No	No
Yes	Yes	No

Table 2-2 shows the complex interactions of anonymous enumerations, coupled with the inclusive membership of the Everyone and Pre-Windows 2000 Compatibility groups.

**Table 2-2 Effect of anonymous enumerations**

Setting			Outcome
Restrict Anonymous Enabled?	Anonymous in Everyone Group	Membership of Pre-Windows 2000 Compatibility Group	Can Anonymous enumerate AD data?
No Effect	No	Empty	No
No Effect	No	Everyone	No
No Effect	No	Anonymous	Yes
No Effect	Yes	Empty	No
No Effect	Yes	Everyone	Yes
No Effect	Yes	Anonymous	Yes

Use Tables 2-1 and 2-2 to strengthen your network against malicious enumerations.

### Windows Logon Authentication Best Practice Recommendations

Windows Logon Authentication is important to Windows computer and network security. Follow these best practice recommendations to strengthen your environment:

- Limit authentication to Kerberos and NTLMv2 authentication whenever possible. LM and NTLM authentication protocols are demonstrably insecure. Disable them whenever you can.
- Disable LM Password Hashes (covered in more detail in Chapter 3). By default, LM password hashes are stored in SAM and AD verification databases by default on all Windows versions. This is the password hash that most brute force hacking tools compromise.
- Disable anonymous enumeration when possible.

Most network administrators never give Windows Logon authentication protocols a second thought—users log on and the protocols do all the hard work behind the scenes. However, as stated in this chapter, understanding and configuring authentication protocols is an important part of any administrator's job.

### Remote Authentication Protocols

Remote authentication protocols are used when a user connects to Routing and Remote Access Services (RRAS) or Internet Authentication Service (IAS). The remote authentication protocol choices are:

- EAP
- MS-CHAPv2
- MS-CHAP
- CHAP
- SPAP
- PAP
- Unauthenticated Access

The list goes from strongest at the top to weakest at the bottom. If several authentication protocols are selected, the client can authenticate to the first available protocol that matches their own options. You may need to enable weaker protocols if the connecting clients don't understand the stronger choices. For example, if Macintosh or Unix clients want to connect, they may support only PAP or CHAP protocols. Also, using weaker protocols often requires that reversible encryption be enabled on the principal's account.

For a remote authentication protocol to work from a Windows client to an RRAS server, the three components must be configured with the same protocol(s) and settings. The three components are:

1. Remote client software
2. Remote Access Policy (RAP)
3. Remote access server (RAS)

If all three components don't share the same protocol settings, remote authentication fails. Even after authenticating remotely successfully, the user often must still go through some form of Windows logon authentication, such as LM, NTLM, or NTLMv2.



### **Note**

**RRAS prevents the use of Kerberos.**

Let's consider each of the remote authentication protocols in turn, from strongest to weakest.

### **Extensible Authentication Protocol (EAP)**

The newest and most secure remote authentication protocol is EAP. Several of the many types of EAP protocols are listed below.

- EAP MD5-Challenge (aka EAP-MS-CHAPv2) uses the same handshake protocol as PPP-based CHAP but sends challenges and responses as EAP messages.
- PEAP-EAP-MS-CHAPv2 is easier to deploy than EAP-TLS because user authentication is accomplished with a password-based credential (username and password) instead of certificates or smart cards (as is used in EAP-TLS). When PEAP-EAP-MS-CHAPv2 is used only RADIUS server (if used) is required to have a certificate (which must be trusted by client).
- PEAP-EAP-TLS uses certificates for server authentication and certificates or smart cards for user and computer authentication. Must use PKI. EAP-TLS can only be used in domain environments running RRAS.

Windows 2000 supports only MD5 and EAP-TLS EAP types. The PEAP protocol layer is new in Server 2003. It is not a remote authentication protocol (or EAP type) by itself; instead, it is used to create an encrypted TLS channel between the PEAP client and the PEAP authenticator (RRAS or IAS server). This channel is a secure tunnel that can be used in conjunction with other remote authentication protocols (EAP-MS-CHAP or EAP-TLS) to provide secure access. PEAP can be used for 802.11 wireless client computers but is not supported for VPN or other RAS client types.

### Microsoft Challenge Handshake Authentication Protocol (MS-CHAP)

After EAP (which many legacy clients don't support), Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) is the most secure remote authentication protocol in Windows. As with NTLM, MS-CHAP's version 2.0 is more secure than the first version and should be used when MS-CHAP support is desired. Version 1.0 provides only one-way authentication (client to server) and uses the same cryptographic key for sending and receiving data. Every time a user connects with the same password, the same cryptographic key is generated because it is based on the user's password. It also allows LM authentication. These limitations were solved in MS-CHAP version 2 (MS-CHAPv2). In contrast, MS-CHAPv2 provides these features:

- Two-way, mutual authentication (client to server and server to client)
- Separate cryptographic keys for sending and receiving data
- Different cryptographic keys used for each connection session
- Does not allow LM authentication

MS-CHAP (version 1 or 2) is the only remote authentication protocol that allows password changes during the authentication process. Windows 95 needs the DUN 1.3 upgrade to support MS-CHAPv2, although it can only be used for VPN connections (not dial-up). Encryption expert Bruce Schneier has written an excellent paper comparing MS-CHAP and MS-CHAPv2 (<http://www.schneier.com/paper-pptpv2.html>).

### Challenge Handshake Authentication Protocol (CHAP)

Challenge Handshake Authentication Protocol (CHAP) was one of the first industry standard protocols to utilize a challenge-response algorithm instead of transmitting authentication credentials in clear text. CHAP is documented in RFC 1334 (<http://www.faqs.org/rfcs/rfc1334.html>), along with its less-secure cousin, PAP (covered below).

After the initial physical link is established, CHAP is used to authenticate the user to the remote authenticator. The CHAP authentication sequence is submitted upon the initial connection and can be requested again at any time during the communication session. A CHAP authentication sequence involves three steps:

1. The authenticator sends a "challenge" to the client.
2. The client responds using a value calculated using their MD5 password hash. For this reason, CHAP authentication requires passwords that are stored with reversible encryption (a security setting disabled by default in Windows 2003 Server).
3. The authenticator checks the client's hashed response against the expected hash calculation. If the two hashes agree, the client is authenticated. CHAP provides limited protection against replay attacks, but cannot be used in Microsoft Point-to-Point (MPPE) tunneling.

### Password Authentication Protocol (PAP) and Shiva Password Authentication Protocol (SPAP)

Both Password Authentication Protocol (PAP) and Shiva Password Authentication Protocol (SPAP) are legacy protocols intended to add a basic user authentication layer to serial access networks when few authentication standards existed. PAP is an open standard, but SPAP is a proprietary protocol implemented by a company with a popular (in the 1990's) line of remote communication devices and modems. Both protocols are often considered together because they had similar features and were released in the same time period. Except for unauthenticated access, PAP and SPAP are the least

secure methods of authentication because the user's credentials are transmitted in clear text and they lack anti-replay mechanisms. When either is used and the initial link is established, the client begins sending the plaintext user ID and password combination repeatedly to the authenticator until it acknowledges the authentication or terminates the connection. If the authentication is accepted, the user's credentials are never passed again or verified. You cannot use Microsoft Point-to-Point (MPPE) tunneling with SPAP or PAP.

### Unauthenticated Access Authentication

Unauthenticated access means the user name and password are not required for remote connection establishment. Unauthenticated access might be useful in mixed-client environments, guest logons, or with DNIS or ANI/CLI authentication. Unauthenticated access may even be desirable, if you offset the risk by implementing some other external form of authentication or security. For example, if the user has already authenticated to the hardware-based VPN solution, it may be reasonable for the remote authentication protocol to be essentially disabled.

### **Internet Information Service (IIS) Authentication Protocols**

Internet Information Service (IIS) has its own set of authentication protocols. You can use them as another layer on top of the NTFS permissions and authentication encountered while accessing underlying files and folders of a web site. Internet Information Service (IIS) authentication protocols include:

- Anonymous
- Basic
- Digest
- Advanced Digest
- Integrated Windows Authentication
  - NTLM
  - Kerberos
- .NET Passport
- SSL/TLS Digital Certificates



#### **Note**

**Share permissions do not apply to IIS-contacted files and folders.**

IIS authentication protocols allow the web administrator to prompt the user, or the user's browser, for a username and password associated with an account either in the local SAM or the Active Directory. You can simultaneously support multiple forms of authentication. IIS will always attempt the more complex or restrictive form of authentication, falling back until it finds a method the client can support.

Authentication protocols can be enabled at the server level, per web site, per folder, and even per file. Like most security permissions in Windows, higher-level authentication protocols inherited above can be overwritten by lower-level authentication protocols that are explicitly set. For example, a web site configured for anonymous access can require that logon credentials be used to access particular folders, web pages, or files.

## Anonymous Authentication

When a web resource is set up with anonymous authentication, neither the user nor the user's browser will be prompted for login credentials. Access is granted using the rights and permissions of the user account affiliated with the IIS anonymous user (the default is IUSR\_computername). There can be only a single IIS anonymous user account shared among all web sites with anonymous authentication enabled. Anonymous enumeration is used where non-authenticated public access is desired.



### Note

**Do not confuse the anonymous user account that IIS uses with the anonymous context used for null session Windows logons.**

## Basic Authentication

Basic authentication prompts the user for a logon name and password, but it doesn't require Internet Explorer or Active Directory. It is used when the web site needs to support a wide variety of Web browsers and clients (such as Mac, OS/2, Unix, and Linux). It uses passwords obscured only with Base64 encoding, which is nearly plain-text. You can improve the security of basic authentication by combining it with SSL.

## Digest Authentication

Digest authentication prompts the user for a login name and password and requires Active Directory running on Microsoft Windows 2000 Server or later. The protocol uses password hashes, not the actual password itself, and it supports authentication through firewalls and proxies. Digest authentication requires you to encrypt and store the user's password with reversible encryption and it requires the connecting client to use Internet Explorer version 5.0 or later.

If Digest authentication is used on a W2K3 web server running in a W2K3 domain, Digest Authentication becomes Advanced Digest Authentication. Advanced Digest was introduced in Windows Server 2003 as an improvement over Digest authentication. It stores a version of the password that has been hashed with MD5. It is available only after the Windows Server 2003 schema extension.

## Integrated Windows Authentication

Integrated Windows authentication is useful for times when you need a moderate level of user authentication, but you don't necessarily need the user to do a manual logon. For example, it is often enabled on intranet web servers; it passes the user's Windows authentication logon credentials to the web site without prompting the user (a setting that can be configured in Internet Explorer). NTLM and Kerberos are both usable in Integrated Windows Authentication. NTLM requires Internet Explorer 2.0 or later, and Kerberos requires Internet Explorer 5.0 or later, or Windows 2000 or later.

## .NET Passport

.NET Passport authentication can be used starting with Windows Server 2003 and IIS 6.0. .NET Passport is essentially another TTP service. A central web site administered by Microsoft performs

authentication, and authentication success is sent to participating web sites. When the user logs on to a web site that requires a .NET passport logon, the user is temporarily redirected to Microsoft's .NET Passport authentication service. The user then authenticates to the .NET Passport service. Participating Web sites never receive a member's password.

The central .NET Passport server(s) return encrypted sign-in and profile information to the participating web site, which can then use the client's Passport authentication approval to write local cookies. The use of cookies avoids redirects back to the central .NET Passport servers on subsequent page views. The .NET Passport central server does not authorize or deny access to individual .sites. It is the responsibility of the Web site to control user access rights.

Microsoft .NET Passport uses standard Web technologies, such as SSL, HTTP redirects, cookies, JScript, and symmetric key encryption. .NET Passport is compatible with Internet Explorer 4 and later, Netscape Navigator version 4.0 and later, and some versions of UNIX.

### **SSL/TLS Digital Certificates**

Web servers or clients can also require the use of trusted digital certificates for authentication. In the typical web server SSL scenario, the client connects to an SSL server; the SSL server then sends the client its public digital certificate, which authenticates the server to the client. The client uses the server's digital certificate to create a secure channel, over which is passed a shared symmetric key that is used to encrypt and decrypt protected communications. Although digital certificates are more likely to be used for server-to-client authentication, the server can require clients to use digital certificates and authenticate, too.

Table 2-3 summarizes IIS authentication protocols and their characteristics.

**Table 2-3 IIS authentication protocol characteristics**

Method	Security Level	How Passwords Are Sent	Crosses Proxy Servers and Firewalls	Client Requirements
Anonymous authentication	None	N/A	Yes	Any browser
Basic authentication	Low	Base64 encoded clear text	Yes, but sending passwords across a proxy server or firewall in clear text is a security risk because Base64 encoded clear text is not encrypted.	Most browsers
Digest authentication	Medium	Hashed	Yes	Internet Explorer 5 or later
Integrated Windows authentication	High	Hashed when NTLM is used; Kerberos ticket when Kerberos is used.	No, unless used over a PPTP connection	Internet Explorer 2.0 or later for NTLM; Windows 2000 or later with internet Explorer 5 or later for Kerberos
.NET Passport authentication	High	Encrypted	Yes, with an SSL connection	Internet Explorer and Netscape
SSL/TLS Digital Certificates	High	n/a	Yes	Most browsers support SSL or TLS

When a user accesses an IIS web site, IIS authentication is performed before access is given. After remotely authenticating to the IIS server, the user's effective permissions are determined by the web site's intersection of NTFS permissions and its IIS permissions (e.g. Read, Execute, Directory Browsing).

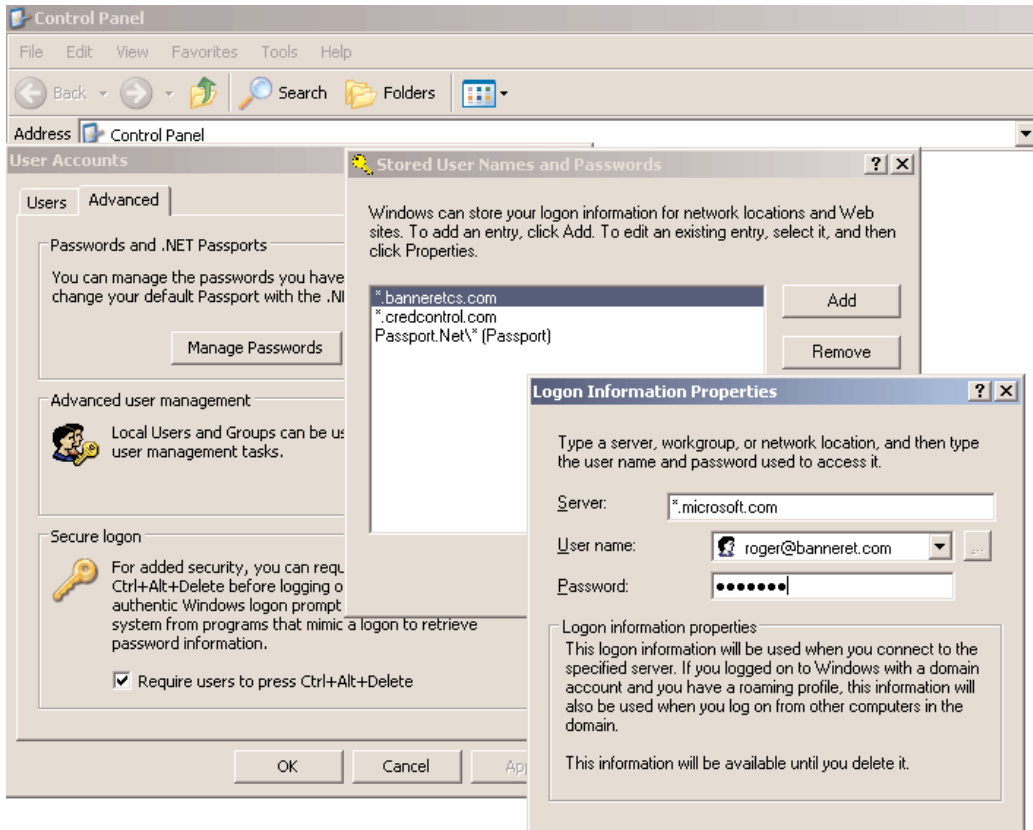
## Microsoft Credential Manager

Microsoft Credential Manager (MCM) is a single-signon solution available in Windows XP and Server 2003. You may have used it without even realizing it, because no GUI is entitled Microsoft Credential Manager. Essentially, MCM gives users an opportunity to store and automatically use passwords that belong to different DNS or NetBIOS domains if the contacted applications are MCM-aware.

If a user or the user's system is prompted for logon credentials, MCM first replies with the user's current logon credentials. If that fails and the application is MCM-aware, the application can prompt MCM for the appropriate credentials, assuming the credentials have been stored before. If the credentials haven't been stored before, the user is prompted for the credentials and be given the opportunity to save the credentials at the Remember my password check box. The credentials are then saved in the user's local or roaming profile and can be access by any application using the MCM API. The next time a user access the same resource, MCM automatically offers the application the mapped credentials without prompting the end user.

You can access the MCM User Interface by opening the Control Panel User Accounts applet, clicking the Advanced tab, and choosing the Manage Passwords option (Figure 2-5).

**Figure 2-5**  
*Microsoft Credential Manager user interface*



### Note

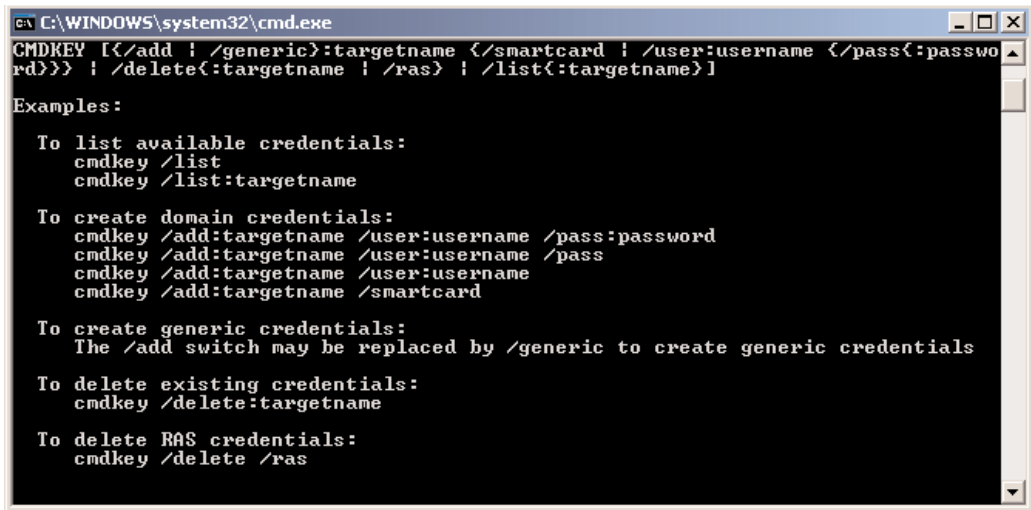
**MCM must be used with caution. The authentication credentials stored by MCM are not synchronized with normal password mechanisms. Therefore, if a stored password is changed or reset on the application side and the MCM is not changed, too, the result is failed logons, and perhaps an account lockout.**

Many security experts believe a single sign on is equivalent to a single failure point—and they're right. If the Windows logon authentication account is compromised, an intruder can then access any sites that are listed in the MCM User Interface (see Figure 2-5) without being prompted for an authentication password.

MCM passwords can be stored and accessed at the command prompt using the Net Use /Savedcred command (<http://support.microsoft.com/?kbid=287536>) in both Windows 2003 and XP. Windows 2003 also has a command prompt utility called CMDKEY (see Figure 2-6) that can be used to manage MCM credentials.

**Figure 2-6**

*CMDKEY command-line options*



```

C:\WINDOWS\system32\cmd.exe
CMDKEY [(</add | /generic>:targetname [</smartcard | /user:username [</pass:<:password>]
rd>>] | /delete:<:targetname | /ras> | /list:<:targetname>]

Examples:

To list available credentials:
cmdkey /list
cmdkey /list:targetname

To create domain credentials:
cmdkey /add:targetname /user:username /pass:password
cmdkey /add:targetname /user:username /pass
cmdkey /add:targetname /user:username
cmdkey /add:targetname /smartcard

To create generic credentials:
The /add switch may be replaced by /generic to create generic credentials

To delete existing credentials:
cmdkey /delete:targetname

To delete RAS credentials:
cmdkey /delete /ras
  
```

## Summary

In this chapter, we took a detailed look at Windows authentication protocols and made best practice recommendations. For strong security, it is essential that Windows authentication be configured correctly. Administrators should disable LM and NTLM protocols, disable anonymous enumerations, and disable LM password hash storage. Unfortunately, even strong authentication practices can be undermined by weak passwords.

Chapter 3 will review password policies and make best practice recommendations for a strong password system.